MASTER EN SCIENCES INFORMATIQUES

Méthodologie de spécification d'une interface homme-machine

Ruche, Jean-Claude

Award date:
1988

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
NAMUR

Institut d'Informatique

Méthodologie de spécification d'une interface homme-machine

Jean-Claude Ruche

Mémoire présenté en vue de l'obtention du titre de
Licencié et Maître en Informatique

1987 - 1988

Je tiens à remercier tout particulièrement le professeur François Bodart pour les conseils judicieux qu' il a prodigués tout au long de l'élaboration de ce travail. Je souhaite également exprimer ma gratitude à Geneviève Warnant pour les remarques pertinentes qu'elle a pu formuler et à Joelle Coutaz pour le temps qu'elle a bien voulu m'accorder lors de mon stage à Grenoble.

Je remercie également tous mes amis et amies, pour le soin qu'ils ont apporté à mes moments de relaxation.

A mon épouse, pour sa patience, sa compréhension et le soutien qu'elle m'a apporté tout au long de l'élaboration de ce travail.

Enfin, un grand merci à mes parents, sans qui cet ouvrage n'aurait jamais vu le jour.

Abstract

Ce travail propose et évalue une méthodologie de spécification d'interfaces-utilisateur. Elle constitue une extension de la méthodologie de spécifications fonctionnelles proposée par F. Bodart et Y. Pigneur pour la prise en compte de l'interaction homme-machine. Une interface peut se décomposer en deux éléments : les fonctionnalités et l'interface. Cette dernière sera spécifiée en terme de Tâche et de Présentation. L'évaluation de la méthodologie nous permet de poser certains des problèmes auxquels sont confrontés les concepteurs d'applications interactives.

This work presents and evaluates a methodology for the specification of User Interfaces. It's an extension of the Requirements Specification methodology proposed by F. Bodart and Y. Pigneur to capture man computer interaction. An interactive system can be divided in two components : the functionnalities and the user interface. This latter will be specified in terms of Task analysis and Presentation. The evaluation of the methodology allow us to discuss about some of the problems that a designer of interactive systems has to face.

Table des matières

Table des figures

Introduction

Il y a une dizaine d'années, l'informatique était réservée aux seuls initiés, aux spécialistes en la matière. De nos jours, la situation s'est bouleversée; l'informatique a conquis les coins les plus reculés de la vie professionnelle et s'attaque à présent à notre vie privée.

Tandis qu'il était important de concevoir des programmes réalisant correctement leurs spécifications, leurs fonctionnalités, l'accent s'est à présent élargi à un tout autre domaine. Bien sûr, il est toujours crucial de fournir un logiciel qui effectue correctement la tâche pour laquelle il a été conçu, mais il faut également qu'il soit convivial, qu'il rencontre les désirs de l'utilisateur, qu'il lui soit adapté. Il n'est plus admissible qu'une personne fasse d'incommensurables efforts pour comprendre l'ordinateur : c'est à ce dernier de se plier à l'utilisateur. Le composant d'une application interactive chargé de réaliser cet objectif ambitieux s'appelle l'interface homme-machine.

Cependant, le problème est vaste et nécessite inmanquablement une approche pluridisciplinaire. Les connaissances nécessaires tiennent de l'art graphique, de la compréhension des capacités motrices, perceptuelles et cognitives de l'être humain, de la connaissance de la technologie en matière d'affichage de données, de techniques d'interaction ou encore de méthodologies de conception. Le tout doit encore être correctement assaisonné d'un petit coup de baguette magique pour en faire une réussite d'élégance et de convivialité!

L'objectif de ce travail est de fournir au concepteur d'application interactive une méthodologie pour lui permettre d'atteindre son but plus aisément. Plus précisément, notre but est d'étendre la méthodologie de spécification proposée par F. Bodart et Y. Pigneur (1983) de manière à prendre en compte, en plus des fonctionnalités, la spécification de l'interface homme-machine. Les concepteurs disposeront alors de toutes les informations pertinentes à la réalisation de leur application dès la phase de spécification. L'importance des spécifications dans le cycle de vie d'un projet informatique est connue de tous. Soulignons que le besoin de prototypage s'affirme encore lorqu'il s'agit d'une application interactive car le concept d'itération est primordial en vue de rencontrer les besoins du ou des futurs utilisateurs.

Cette méthodologie pourra s'appuyer ultérieurement sur un outil d'aide à la conception tel qu'un Système de Gestion de Dialogue pour permettre la construction de l'application interactive à partir des spécifications.

Notre exposé précisera au chapitre I le contexte dans lequel notre travail s'est effectué. Nous recenserons les problèmes majeurs qui se posent lors la construction d'applications interactives.

Le chapitre II répertoriera pour sa part les outils qui existent à l'heure actuelle pour aider le concepteur dans sa tâche de création et d'implémentation d'une application interactive. Nous verrons que ces outils doivent encore être améliorés pour constituer une aide véritable.

Le chapitre III détaillera la modélisation et l'architecture d'une application interactive sur lesquelles nous nous sommes basés pour développer notre méthodologie. Elles s'appuient sur le principe de séparation modulaire entre les fonctionnalités et l'interface.

Le chapitre IV présentera en détails la méthodologie. Rappelons que son but est d'étendre la méthodologie de spécification fonctionnelle que nous utilisons à l'Institut d'Informatique Notre-Dame de la Paix. Elles se décomposera en deux étapes : la spécification de la Tâche et la spécification de la Présentation.

Les chapitres V et VI proposent une expérimentation de la méthodologie. Elle se basera sur un exemple bien précis de saisie interactive d'une commande. L'énoncé et les spécifications fonctionnelles de l'exemple seront présentés au chapitre V. Le chapitre VI procédera à la spécification de l'interface et à son évaluation. Il mettra en évidence les remarques qui ont pu être faites lors de la réalisation de l'application interactive.

Le chapitre VII analysera les avantages et insuffisances de notre esquisse méthodologique de manière à pouvoir ultérieurement approfondir et préciser son contenu.

L'annexe 1 présente informellement l'exemple à la base de notre expérimentation. L'annexe 2 constitue les spécifications que nous avons réalisées pour son implémentation sur Turbo/Pascal. Enfin, l'annexe 3 expose et évalue son implémentation à l'aide d'Hypercard.

Chapitre I :
Historique et contexte des applications interactives

I.0. Introduction

La conception d'une application interactive n'est pas une tâche simple. Les problèmes auxquels sont confrontés les concepteurs trouvent leur solution dans diverses disciplines ayant toutes un rôle à jouer dans la réalisation d'une application interactive efficace.

Ce chapitre a pour but de situer quelques uns des aspects de la problématique. Nous n'essaierons pas de tout passer en revue mais bien de donner au lecteur une idée des problèmes qui peuvent se poser.

Nous partirons tout d'abord d'une brève perspective historique pour ensuite définir quelques concepts que nous utiliserons tout au long de ce travail. Nous aborderons alors le vaste problème de l'utilisateur et de l'utilisation d'une application interactive qui peut être analysé sous différents angles de vue comme l'ergonomie ou la psychologie cognitive. Nous citerons les techniques d'interaction les plus connues en les regroupant par styles et, enfin, nous introduirons certains principes et méthodologies qui existent pour aider le concepteur dans la réalisation de sa tâche.

Il est intéressant de faire un bref retour dans le passé pour constater que les fondements des systèmes interactifs furent posés dans les années 60 et 70. C'est l'objectif de la section suivante.

I.1. Perspective historique

Dès les années 50, certaines personnes entrevirent les potentialités de l'ordinateur en temps qu'aide à la créativité humaine et à la prise de décision. Parmi eux, J. C. R. Licklider [Licklider,1960] imagina une synergie entre les capacités de la machine et celles de l'homme, ce qu'il appela la "symbiose homme-machine".

Avec l'avènement de l'informatique conversationnelle [Davis, 1966; Fano et Corbato, 1966; Licklider, 1968] et l'apparition des réseaux [Roberts, 1986], l'importance de l'interaction homme-machine s'affirma.

A la même époque, des expériences eurent lieu concernant la communication graphique entre l'homme et la machine [Sutherland, 1963] et des recommandations furent émises pour la réalisation d'un système d'aide à la conception [Coons, 1963].

Il fallut attendre des progrès technologiques en matière de graphisme pour voir la naissance de nouvelles applications dans des domaines tels que les mathématiques, la science, l'art... Les types d'applications informatiques qui eurent le plus grand impact furent le traitement de texte et l'aide informatique à la conception de documents [Engelbart, 1963, 1982, 1986; Nelson, 1965, 1973, 1974, 1981].

Les premières expériences sur la qualité de l'interface eurent lieu dans le milieu des années 60. Le premier à parler de méthodes de recherche psychologique concernant la programmation fut Sackman [Sackman, 1970]. Les psychologues et les spécialistes des facteurs

humains se penchèrent dès lors sur la question de l'interaction homme-machine [Weinberg, 1971].

Un évènement marquant dans la popularisation et la consolidation des problèmes d'interaction homme-machine fut le livre de James Martin (1973) "Design of Man-Computer Dialogues".

Au niveau du développement technologique et logiciel, il faut noter la concentration d'informaticiens de talent dans le milieu des années 70 au centre Xerox. Elle engendra le développement d'un prototype de station de travail personnelle [Thacker et al, 1979], avec une interface graphique [Lampson, 1986] et une connection réseau [[Metcalfe 1976].

L'intelligence artificielle eut également son rôle dans le développement d'une interface homme-machine dite intelligente. Nous reportons le lecteur à la section II.3.4. pour plus de renseignements.

Un phénomène essentiel fut l'apparition des "Personal Computers" qui permit la popularisation de l'ordinateur. L'informatique n'était plus réservée aux seuls techniciens et spécialistes. Parmi toutes les classes d'utilisateurs, un désir commun était formulé : le besoin d'une interface plus conviviale, plus souple, plus "user-friendly".

L'importance de ce besoin peut être perçu par l'enthousiasme qui accueillit le Macintosh d'Apple [Williams, 1984], le premier P. C. conçu avec une interface du style développé par Xerox.

Les études qui furent faites sur la modélisation du dialogue prirent réellement leur essor dans les années 1970. Nous consacrerons le chapitre III et une partie du reste de ce chapitre à cet effet.

Pour conclure ce bref historique, nous insistons sur l'impératif économique actuel de réaliser des applications interactives rencontrant les besoins et les désirs des utilisateurs. Une application interactive ne rencontrant pas cette contrainte est vouée à un échec à court ou moyen terme.

Pour le lecteur intéressé par plus de détails sur la perspective historique de l'interaction homme-machine, nous conseillons particulièrement [Baecker, 1987] ainsi que [Strohmeieir, 1986; Bergerot, 1971].

Nous allons à présent définir quelques notions qui nous seront utiles dans la suite de ce travail.

I.2.Définitions

I.2.1. Interface-utilisateur

Une interface au sens informatique du terme est l'ensemble des règles et conventions qui régit la communication entre deux systèmes organisés [Chernicoff, 1985].

En ce qui nous concerne, l'interface homme-machine règle le dialogue entre l'utilisateur et l'ordinateur [Petoud, 1986]. Dans le but de faciliter la tâche de l'utilisateur, il est nécessaire de concevoir une interface qui se rapproche le plus possible du mode de pensée et du langage de l'utilisateur. Nous définirons plus loin (sections I.2.3 et I.3.) certains éléments susceptibles de clarifier cette exigence.

Remarque : dans la suite de ce travail, interface homme-machine, interface-utilisateur et dialogue seront utilisés en tant que synonymes.

I.2.2. Application interactive

Au sens strict du terme, une application interactive implique l'intervention de l'opérateur dans le traitement qu'il effectue. Par opposition, les systèmes "en temps différé" ou systèmes "batch" ne nécessitent pas l'intervention de l'utilisateur. Une application interactive doit être une aide à l'utilisateur dans la réalisation de sa tâche [Faulle, 1982].

Plus généralement, on parlera d'une application interactive lorsqu'elle est caractérisée par la part importante de dialogue qu'elle nécessite avec l'utilisateur pour assurer l'exécution des fonctionnalités qu'elle offre [Chandelon & Warnant, 1987].

Remarque : Nous parlerons indifféremment d'application ou de système interactif.

I. 2. 3. User-friendliness

Le concept de "user-friendliness" est vague - un tel ordinateur ne fatigue pas l'utilisateur, ne détruit pas aléatoirement son travail, lui permet d'effectuer son travail avec succès, explique ses propres actions... [Böller, 1987]

Pour remédier à ce fait, certains auteurs ont essayé d'isoler des critères d'évaluation de la qualité d'une interface :

* Une application interactive doit être [Dzida, Herda & Itzfeldt, 1979] :

- auto-descriptive,
- contrôlée par l'utilisateur,
- facile à apprendre,
- adéquate au problème posé par la tâche,
- en correspondance avec l'attente de l'utilisateur,

- flexible dans l'accomplissement de la tâche,
- tolérante aux fautes.

* Schneiderman (1987) propose 5 critères pour évaluer la qualité de l'application interactive :

- le temps d'apprentissage,
- le taux de performance,
- le taux d'erreur des utilisateurs,
- la satisfaction subjective,
- la rétention à travers le temps.

L'avantage des critères proposés par Schneiderman est la possibilité, une fois l'importance relative de chaque critère établie, d'obtenir des résultats quantitatifs de l'application testée et d'en informer les clients ou les utilisateurs ainsi que de guider les concepteurs et les implémenteurs.

Après ces quelques définitions, nous allons nous attarder quelque peu à l'utilisateur et l'utilisation d'application interactive.

" *Understanding the physical, intellectual and personality differences among users is vital*" [Schneiderman, 1987].

L'objectif de cette section n'est pas de reprendre tous les éléments qui jouent un rôle dans le vaste domaine consacré à l'analyse de l'utilisateur et de son interaction avec l'ordinateur, mais bien de présenter au lecteur les grands domaines dans lesquels s'effectuent les recherches en la matière.

Nous débuterons par une tentative de classification des utilisateurs, pour ensuite analyser, selon une approche ergonomique, l'environnement physique dans lequel se déroulera l'application. Nous nous attarderons enfin aux aspects psychologiques de l'utilisation d'un ordinateur.

I.3.1. Classification des utilisateurs

La diversité des utilisateurs a deux incidences profondes sur la conception d'applications interactives : l'obligation d'avoir recours à une classsification des utilisateurs selon divers critères tels que la connaissance liée à la tâche, la connaissance liée à l'outil[1], l'appartenance à un groupement professionnel... et le besoin d'indépendance entre les fonctionnalités de l'application interactive et son interface-utilisateur, afin de permettre la réalisation de plusieurs interfaces pour une même application interactive. Ce dernier point fera l'objet de la section I.5.1.

Une classification des utilisateurs est indispensable vu l'impossibilité actuelle de réaliser une interface adaptée à un utilisateur bien particulier. L'intelligence artificielle tente cependant d'y parvenir par la conception d'interfaces-utilisateur "intelligentes". Ces dernières construiront une représentation de la connaissance et des aptitudes de l'utilisateur afin d'y adapter leurs

[1] On fait généralement la distinction entre novice et expert.

comportements (voir section II.3.4.). L'aboutissement de ce projet rendrait alors toute classification inutile.

Le regroupement d'utilisateurs en différentes classes est cependant souvent peu explicite, grossier et confus. Où est la frontière entre un novice et un expert? Il existe un manque d'information quantitative permettant une classification utile. Böller (1982) propose un ensemble de dimensions de connaissance pouvant permettre de différencier les utilisateurs :

- l'éducation,
- l'entraînement,
- la connaissance de la tâche,
- la connaissance de l'outil,
- la pression du temps,
- la motivation d'apprendre.

Ces différentes dimensions peuvent permettre d'évaluer avec un certain degré de précision la connaissance que possède l'utilisateur dans chacune d'elles, pour en permettre ensuite la classification. Cependant, la classification résultante, pour être appropriée, doit nécessairement être complexe. A l'heure actuelle, peu d'application interactive peuvent se targuer de posséder des interfaces-utilisateur adaptées à plus d'une ou deux classes d'utilisateur. Un problème de coût semble empêcher de réels progrès en la matière, malgré les principes et méthodologies (voir section I.5.) et les outils (voir chapitre II) actuels, facilitant la conception rapide d'applications interactives. Une interface intelligente semble la solution la plus prometteuse pour résoudre le problème.

La diversité des utilisateurs peut être analysée et/ou résolue par l'ergonomie ou la psychologie. C'est l'objectif des deux sections suivantes.

I.3.2. Approche ergonomique : l'environnement physique

L'ergonomie s'est penchée la première sur les implications et l'environnement de l'interaction homme-machine. Les facteurs en jeu sont avant tout techniques mais une technologie de pointe dans un mauvais environnement ne donnera que de mauvais résultats.

L'ergonomie nous donne des moyens d'étude des conditions de travail idéales ou suffisantes pour un bon rendement et une satisfaction de l'utilisateur. Les problèmes liés à l'environnement physique peuvent être classifiés en 6 domaines principaux [Baecker, 1987] :

I.3.2.1. Ergonomie du poste de travail

Nous entendons ici par poste de travail le terminal et le matériel employé par l'utilisateur (sa chaise, son bureau...) pour réaliser sa tâche. Ceux-ci doivent être adaptés aux dimensions et aux capacités physiques de l'utilisateur. Deux remarques doivent cependant être faites :

Tout d'abord, il faut savoir qu'un matériel ergonomique peut faire empirer les choses. Les problèmes de maux de dos sont courants chez les personnes passant de longs moments devant leur poste de travail. Des sièges de bureau "ergonomiques" sont à présent commercialisés, offrant à l'utilisateur une quantité de réglages à effectuer pour adapter le siège à

ses propres dimensions. Mais peu d'utilisateurs savent quels sont les réglages qui leur conviennent d'un point de vue ergonomique et, au lieu d'adapter leur siège, le rendent encore moins ergonomique!

Ensuite, un matériel ergonomique n'est pas nécessaire à toutes les catégories d'utilisateurs. Il concerne principalement les personnes assignées à des tâches routinières et répétitives, passant une grande partie de leur temps devant leur poste de travail.

I.3.2.2. Lumière

Le reflet de l'écran est un problème qui se pose fréquemment aux utilisateurs. Il existe des filtres pour résoudre partiellement le problème ou des moyens plus complexes tenant compte de la position de l'écran par rapport aux fenêtres et aux lampes, utilisant des déflecteurs de lumière fluorescente... [Chapman et Knutson, 1985]

I.3.2.3. Température et qualité de l'air

Le passage de simples terminaux à des stations de travail personnelles implique le besoin de gérer le supplément de chaleur et de bruit qu'il entraîne. Des problèmes tels que la sécheresse ou l'humidité de l'air, la température, ont des effets directs sur les performances de l'homme ou de la machine.

I.3.2.4. Bruit

Le son a une influence considérable sur la concentration, le degré de stress ainsi que d'autres aspects liés aux performances de l'utilisateur. Cependant, cette influence est complexe et sujette à un grand nombre d'études donnant parfois des résultats contradictoires.

I.3.2.5. Santé

L'introduction de toute technologie entraîne des considérations de santé sur son utilisation. On s'est rendu compte que les problèmes de santé était souvent liés à une mauvaise utilisation de la technologie plutôt qu'à la technologie en elle-même. Un besoin d'apprentissage semble nécessaire pour les éviter.

I.3.2.6. Standard

Des standard concernant l'ergonomie du poste de travail ont été émis à des niveaux nationaux ou internationaux. Malheureusement, ces standard sont parfois contradictoires et risquent d'être appliqués en dehors de leur contexte. Il est important en cela de bien comprendre les problèmes posés avant de légitimiser des standard prématurément.

I.3.2.7. Conclusion

Voici brièvement esquissés les problèmes qui peuvent se poser au niveau de l'environnement physique de l'utilisateur. Les théories foisonnent en ce domaine. Un juste compromis doit cependant être fait entre ces théories et la situation réelle. Pour illustrer ce besoin, citons une étude qui a été faite sur la position des utilisateurs à leur poste de travail. Elle a révélé que la plupart des utilisateurs se penchaient légèrement plus en arrière que ce que l'on

considérait comme une position ergonomique. Or, la conception des écrans et des claviers avait été réalisée avec l'hypothèse de base qu'un utilisateur se tenait selon la position ergonomique! Cette simple constatation signifiait que ni les écrans, ni les claviers ne rencontraient leurs objectifs ergonomiques.

L'ergonomie s'intéresse avant tout aux problèmes d'environnement. La psychologie peut aider le concepteur d'application interactive à dégager les modes de pensée de l'utilisateur et à adapter en ce sens son application.

I.3.3. Aspects psychologiques

L'efficacité et l'acceptation d'une application interactive dépendent de la qualité de la communication entre son interface et l'utilisateur. Un dialogue de qualité permettra une rapide compréhension et un apprentissage aisé des opérations nécessaires à l'accomplissement de la tâche. Le concepteur dispose de deux moyens pour parvenir à l'élaboration d'une application interactive réalisant cet objectif :

Le premier est de procéder à une étude empirique de l'application réalisée en vue d'améliorer le produit par itérations successives; il s'agit alors d'effectuer des mesures empiriques de performance d'utilisation et d'apprentissage pour comparer les différentes interfaces-utilisateur réalisées. Mais l'approche empirique souffre de sérieuses limitations car il est virtuellement impossible de tester ne fût-ce qu'un petit ensemble des variations possibles d'implémentation d'une application interactive.

La deuxième méthode tente de remédier aux carences de la première; il s'agit de développer une famille de modèles analytiques pour prédire quantitativement les performances de l'utilisateur avant même que l'application interactive ne soit construite. Grâce à ces modèles, il sera possible de minimiser le besoin de construire de multiples versions et de produire de longues expérimentations.

Une grande partie de ces modèles provient de la psychologie moderne, plus particulièrement de la psychologie cognitive ou de la psychologie de traitement de l'information [Lindsay & Norman, 1977; Gardner, 1987]. Ils permettent une analyse sinon quantitative dans le meilleur des cas, du moins qualitative entre plusieurs choix de conception.

Ces modèles ne sont que les prémisses d'une discipline qui devrait voir son rôle accru dans les prochaines années. Ils manquent souvent de généralité et ne capturent que les aspects les plus simples du processus cognitif de l'être humain.

Notre but n'est pas de reprendre en détails les différents travaux réalisés sur le sujet mais simplement d'introduire les études les plus connues, en donnant les références adéquates au lecteur intéressé. Parmi les ouvrages reprenant ces modèles, nous pouvons citer [Coutaz, 1986a; Baecker, 1987; Schneiderman, 1986; Böller, 1982].

I.3.3.1. L'homme en tant que processeur d'information

Le modèle du processeur humain [Card, 1983] est un modèle idéalisé de traitement de l'information. L'homme est modélisé en tant que système d'information. Ce système est organisé en 3 sous-systèmes interdépendants : les systèmes moteur, perceptif et cognitif.

Chaque système comprend un processeur et une mémoire. Les processeurs et les mémoires sont caractérisés par des paramètres tels que le cycle de traitement, la capacité mémoire,...

Ce modèle permet de prédire les performances de l'homme dans des domaines aussi variés que la perception, les capacités motrices, les décisions simples ou l'apprentissage, domaines ayant une importance considérable dans l'interaction homme-machine. Par exemple, il permet d'étudier comparativement la vitesse d'un utilisateur avec différents appareils de manipulation directe ou de claviers. Pour cela, on évalue les paramètres qui caractérisent la performance à étudier à l'aide de formules ou de lois telles que le temps pour positionner sa main sur un objet de taille S à une distance D de la main : $t = I \log2(D/S + 0.5)$ [Coutaz, 1986a]. Le modèle du processeur humain est à la base du modèle suivant.

I.3.3.2. L'analyse de tâches cognitives routinières

Le modèle du processeur humain essaie de formaliser les procédures qu'effectue une personne lorsqu'elle mène à bien certaines tâches en interaction avec un ordinateur. Card, Moran et Newel (1980b) montrent qu'il est déjà possible d'établir ainsi des modèles possédant une certaine puissance prédictive, malgré leur côté préliminaire.

Le modèle qu'ils ont développé est connu sous le nom de GOMS (Goal, Operator, Method, Selection rule). Il porte sur une tâche qu'ils considèrent comme routinière, à savoir l'édition de texte. C'est une représentation de la structure cognitive de l'utilisateur en terme de buts, d'opérateurs, de méthodes pour atteindre les buts et de règles de sélection permettant de choisir parmi plusieurs méthodes. Les résultats portent à la fois sur une prédiction des méthodes que l'utilisateur emploie pour réaliser certaines tâches et sur le temps qu'il lui faudra pour l'accomplir.

De plus, l'étude est réalisée à différents niveaux de détail : au niveau de l'unité de la tâche, au niveau fonctionnel, au niveau de l'argument et enfin au niveau de la frappe d'une touche du clavier (the Keystroke-Level Model).

Ce dernier, qu'on peut appeler le niveau lexical, a fait l'objet d'une étude détaillée [Card, 1980a] afin de prédire les performances de l'utilisateur au niveau du temps nécessaire à la réalisation de la tâche. Le modèle permet de calculer dans ce but le nombre de touches-clavier que l'utilisateur devra taper ainsi que le temps de préparation mentale et le temps de réponse du système.

Mais de sérieuses limitations restreignent le champ d'applicabilité du modèle. On suppose en effet que l'utilisateur ne commet pas d'erreur et qu'il est un expert dans la réalisation de sa tâche. Ces hypothèses sont assez importantes quand on sait qu'il est plus difficile de concevoir une application interactive pour des novices et que des erreurs sont et seront toujours commises. De plus, le modèle s'applique à des applications de traitement de texte et ne propose pas de généralisation à d'autres types d'applications.

L'importance de ce modèle reste cependant considérable car, bien que dans un domaine fort limité, il permet une étude quantitative des performances de l'interface qui sera réalisée.

I.3.3.3. Grammaire de langage de commande

Moran (1981) formula la "Command Language Grammar" décrivant l'interface utilisateur d'un système interactif en 6 niveaux :

- **task level :** l'ensemble des tâches devant être accomplies par l'application interactive,

- **semantic level** : la spécification des objets conceptuels manipulés par l'application interactive et les actions conceptuelles que l'application interactive peut accomplir,

- **syntactic level** : la définition du langage de commande à l'aide duquel l'utilisateur communique avec l'application,

- **interaction level :** les actions physiques associées à chaque élément du niveau syntaxique,

- **special layout level** : la description de l'arrangement des appareils d'entrée/sortie et de l'affichage graphique,

- **device level** : la description de tous les autres aspects physiques.

Chaque niveau est un raffinement du précédent. Contrairement aux deux premiers modèles servant à analyser les performances de l'utilisateur; ce modèle propose en quelque sorte une méthodologie (voir section I.5.2.) de conception d'une application interactive basée sur une validation psychologique en ce sens où elle permet de reflèter la structure linguistique et psychologique de l'individu.

I.3.3.4. Théorie de l'action et modèles conceptuels

Cette théorie porte sur les performances de l'accomplissement d'une tâche. La base de la théorie de D. Norman (1984, 1985, 1896) s'appuie sur l'hypothèse que l'utilisateur élabore des modèles conceptuels de sa tâche lorsqu'il est en interaction avec un ordinateur. Un modèle conceptuel est *"une représentation mentale de soi-même et de l'environnement; il dépend de la connaissance et de la compréhension préalable; il est modifié par la nature de l'interaction"* [Coutaz, 1986a]. Norman distingue le modèle conceptuel de l'utilisateur, celui du designer et, éventuellement, de l'ordinateur[2]. Il définit de même l'image du système présenté à l'utilisateur (ce qu'il voit à l'écran).

L'accomplissement d'une tâche se réalise au moyen de séquences variées, choisies parmi 7 étapes :

- établir un but (la représentation mentale de l'état du système que l'on veut atteindre),
- former une intention (l'évaluation de la distance entre le but et l'état courant du système),
- spécifier une séquence d'actions (traduction de l'intention en actions physiques) ,
- exécuter l'action,
- percevoir l'état du système (traduction de l'état en termes psychologiques),
- interpréter l'état (interprétation de l'état perçu),
- évaluer l'état du système par rapport aux buts et aux intentions.

---

[2] Dans le cas d'un programme "intelligent" (qui s'adapte à l'utilisateur).

Cette théorie ne permet pas d'étude quantitative mais bien qualitative. Cependant, elle met en valeur des aspects intéressants dans le domaine de l'interaction homme-machine. Par exemple, elle révèle l'existence d'un fossé entre la représentation mentale que se fait l'utilisateur et la présentation physique qui lui est offerte. Un objectif de tout concepteur doit être dès lors de combler ce fossé en lui présentant une image du système aussi proche que possible de sa représentation mentale.

I.3.3.5. Théorie de la connaissance : le modèle de la connaissance sémantique/syntaxique [Schneiderman, 1987]

La distinction entre la syntaxe et la sémantique, en informatique, provient des constructeurs de compilateurs voulant séparer la validation du texte en entrée des opérations déclenchées par ce texte.

Le modèle suggère que les utilisateurs ont une connaissance syntaxique des détails dépendant de l'appareil utilisé tandis qu'ils ont une connaissance sémantique des concepts qui s'y retrouvent. La connaissance sémantique est elle-même divisée en concepts liés à la tâche (objets et actions) et en concepts liés à l'ordinateur (objets et actions).

Ce modèle montre que l'utilisateur doit acquérir de la connaissance sémantique à propos des concepts liés à l'ordinateur car la connaissance syntaxique est arbitraire, dépendante du système et sans structure. Elle est acquise par pure mémorisation, par répétition et est vite oubliée. Par opposition, la connaissance sémantique est organisée hiérarchiquement, acquise par apprentissage ou analogie, et indépendante des détails syntaxiques.

B. Schneiderman conseille d'identifier clairement la sémantique des objets de la tâche et des actions de l'utilisateur qui y sont associées . Ensuite, d'identifier les objets et les actions liés à l'ordinateur, indépendamment des détails syntaxiques. De la sorte, la conception qui en résultera sera plus compréhensible pour les utilisateurs et plus indépendante d'un matériel spécifique.

I.3.3.6. Conclusion

Nous n'avons pas cité tous les modèles qui existent sur le sujet. Le but était ici de montrer l'importance qu'il fallait accorder à l'étude de l'interaction homme-machine d'un point de vue cognitif. Ces modèles ont la commune ambition de capter les éléments psychologiques jouant un rôle dans la réalisation d'une tâche. Ils dissocient à cet effet les aspects liés à la tâche (connaissance sémantique, task and semantic levels...) des aspects de présentation (connaissance syntaxique, syntactic, interaction, special layout levels...). Nous adopterons ce principe dans la méthodologie que nous présentons au chapitre III et IV.

Ces théories, modèles ou principes, permettent d'évaluer les systèmes interactifs, de prédire leurs futures performances, en bref, d'aider à une meilleure conception d'applications interactives. Pour reprendre Stu Card : *"Theory adds wings to intuition"*. Cependant, le modèle de Card, Moran et Newell est la première tentative compréhensible vers une théorie quantitative de l'interaction homme-machine. Il reste encore beaucoup à faire. Comme nous l'avons vu, il

existe plusieurs classes d'utilisateurs, mais également différentes dimensions de performance et différentes sortes de tâches. Peu de ces domaines ont été explorés à l'heure actuelle.

Malgré certains modèles prometteurs, la connaissance dans le domaine de l'interaction homme-machine reste faible quant aux aspects psychologiques liés à la résolution d'un problème complexe tel que l'apprentissage ou la mémorisation.

Nous allons à présent aborder les différents styles et techniques d'interaction qu'un concepteur d'application pourra utiliser.

I.4. Styles et techniques d'interaction

Les techniques d'interaction foisonnent parmi les applications interactives. Une manière[3] de les organiser est de les regrouper par style ou famille de dialogue. Un style de dialogue peut être vu comme une interface offrant un ensemble de techniques d'interaction, unifié et cohérent.

Nous présentons 9 catégories principales de style d'interaction [Baecker, 1987]:

- **La ligne de commande** : La méthode la plus traditionnelle pour donner des instructions au système interactif est l'introduction de lignes de commande. L'ensemble des commandes autorisée fait partie d'un langage de commande formel.

- **Le langage de programmation** : L'ensemble des commandes que peut effectuer l'utilisateur peut être tellement complexe et vaste qu'il est nécessaire d'offrir un langage de programmation pour pouvoir avec l'application interactive. C'est le cas par exemple des systèmes d'interrogation de base de données.

- **Le langage naturel** : On qualifie parfois d'artificiel [Perlman, 1984] les langages de commande car ils possèdent une grammaire fort réduite. L'utilisation d'un sous-ensemble bien défini d'un langage naturel tel que le français évite l'apprentissage d'un nouveau langage et favorise son acceptation.

- **Le menu** : L'utilisateur effectue des choix et déclenche des opérations sans qu'aucun langage de commande ne lui soit nécessaire. Ceci est particulièrement intéressant pour les utilisateurs occasionnels ou peu expérimentés.

- **Le remplissage de formulaire** : L'utilisateur envoie des commandes en remplissant un ou plusieurs formulaires affichés à l'écran.

- **L'icône** : La représentantion d'une commande ou d'un "feedback[4]" du système s'effectue sous forme graphique.

---

[3] Il en existe bien d'autres.

[4] Un feedback est une information que le système fournit en réponse à une action de l'utilisateur. Voir section I.5.2.2.

- **La fenêtre** : Une fonction spécifique ou un terminal virtuel est représenté à l'écran par une surface rectangulaire.

- **La manipulation directe** : L'utilisateur manipule, à l'aide d'une souris ou d'un autre mécanisme de sélection, une représentation graphique des données de l'application;

- **L'interaction graphique** : L'utilisateur définit ou modifie des images ou des dessins en 2 ou 3 dimensions.

Une même application interactive utilisera généralement plusieurs de ces styles. Par exemple, on verra l'apparition simultanée d'icônes et de fenêtres avec manipulation directe.

Les performances de chacun de ces styles a été analysé selon le type de tâche et les capacités de l'utilisateur. Les résultats n'ont pas donné d'avantages ou d'inconvénients décisifs pour l'un ou l'autre style. Nous reportons le lecteur à [Schneiderman, 1987] et [Baecker, 1987] pour plus de renseignements.

Après avoir présenté les styles de dialogue qui pouvaient être utilisés par le concepteur d'une application interactive, nous allons à présent analyser quelques principes et méthodologies qui peuvent l'aider dans la réalisation de sa tâche.

I.5. Principes et méthodologies

Le concepteur d'une application interactive dispose d'une énorme variété de technologies et de techniques d'interaction. Il doit alors choisir parmi elles celles qu'il considère comme les plus appropriées à son problème.

Il existe cependant des principes et des méthodologies qui peuvent l'aider dans sa tâche. Le chapitre IV spécifiera en détails la méthodologie que nous proposons dans le cadre de ce travail. Cette section vise à reprendre certains des travaux qui ont été élaborés sur le sujet.

Des principes sont une collection de conseils donnés au concepteur d'une application interactive tandis qu'une méthodologie consiste en des règles plus ou moins formelles et partiellement ordonnées qui ont pour but de faciliter le processus de conception et de conduire à un résultat satisfaisant.

I.5.1. Principes de conception

I.5.1.1. Séparation modulaire entre l'interface et les fonctionnalités

Ce principe découle directement du modèle de la connaissance sémantique/syntaxique développé par Schneiderman. Il s'avère également indispensable vu la diversité des utilisateurs et donc le besoin de réaliser plusieurs interfaces-utilisateur pour rencontrer les caractéristiques des utilisateurs[5].

[5] Cfr section I.3.1.

Cette séparation modulaire permet *"l'indépendance de conception de l'interface par rapport à la conception des fonctionnalités de l'application, l'indépendance de l'implémentation de l'application face au matériel ou aux techniques d'interaction qui seront utilisés, la facilité de réemploi du code de l'application et enfin la construction itérative de prototypes de l'interface"* [Chandelon & Warnant, 1987].

Ce principe de séparation semble admis par la plupart des chercheurs dans le domaine [Coutaz, 1986a, 1986b; Konsynski, 1985; Rosenthal et Sufrin, 1983; Ahlsen et Britts, 1986; Edmonds, 1981; Sproull, 1983; Draper & Norman, 1984].

I.5.1.2. Principes de Hansen (1971)

Une des premières énumérations de principes de conception fut présentée par Hansen (1971). Elle s'adresse à l'élaboration d'applications interactives graphiques :

- *connaître l'utilisateur*

- *minimiser la mémorisation :*
  . permettre la selection plutôt que l'entrée de données
  . utiliser des noms plutôt que des nombres
  . assurer un comportement prévisible de l'application
  . permettre l'accès à des informations utiles

- *optimiser les opérations :*
. exécuter rapidement les opérations routinières
. modifier le moins possible l'affichage pendant l'exécution d'une opération[6]
. organiser et réorganiser les paramètres de commande selon l'usage qui en est fait (adaptabilité)

- *"engineer for errors" :*
. produire de bons messages d'erreur
. concevoir l'application de manière à empêcher les erreurs les plus communes
. permettre des actions réversibles
. offrir de la redondance
. garantir l'intégrité des données en cas de panne software ou hardware.

I.5.1.3. Les 8 règles d'or de Ben Schneiderman (1987)

Les règles que proposent B. Schneiderman s'appliquent à la majorité des systèmes interactifs :

- *viser l'uniformité* : l'utilisation de différentes applications ne doit pas être perturbée par des incohérences entre les interfaces[7];
- *permettre des raccourcis aux utilisateurs fréquents* (par exemple grâce à l'utilisation de "macros"[8]);

---

[6] Tout en montrant à l'utilisateur qu'une opération est en train de se dérouler.

[7] Des noms différents pour la même fonction, un même nom pour des fonctions différentes...

[8] Une suite d'actions que l'utilisateur rassemble pour pouvoir les exécuter en une seule fois.

- *offrir un feedback informatif;*
- *concevoir des dialogues permettant la clôture* : une tâche qui demande une longue concentration fatigue l'utilisateur et de la sorte, diminue ses performances. Diverses étapes doivent échelonner son travail pour lui permettre de relâcher son effort régulièrement;
- *offrir une gestion simple des erreurs* : une erreur ne doit pas avoir de résultats catastrophiques pour l'utilisateur. Il doit pouvoir la corriger aïsément;
- *permettre le retour en arrière aisé* : un utilisateur peut à tout moment désirer de recommencer une partie de son travail, par exemple s'il n'est pas content du résultat. L'application doit lui en fournir la possibilité;
- *offrir à l'utilisateur le sentiment de contrôler l'application* : l'application doit être au service de l'utilisateur et non le contraire. Son comportement devra être conçu en ce sens;
- *réduire la charge de mémoire à court terme.* : un écran affichant une multitude d'informations en même temps risque à la fois de stresser l'utilisateur et lui empêcher d'effectuer correctement sa tâche (il se sentira envahi par toutes ces données).

D'autres listes de principes peuvent être trouvées dans [Baecker, 1987; Gould et Lewis, 1985; Norman, 1983; Rubinstein et Hersh, 1984].

I.5.1.4. Conclusion

Les principes que nous avons énoncés sont avant tout des conseils généraux provenant d'une expérience passée ou encore d'études détaillées ayant débouchés sur des critères à respecter. Ces principes sont importants mais ne nous donnent pas de recette à employer, ils citent seulement les ingrédients. Les méthodologies se proposent de donner la marche à suivre pour réaliser une application interactive rencontrant les objectifs que nous avons introduits.

I. 5. 2. Méthodologies de conception

On peut distinguer 2 niveaux de généralité parmi les méthodologies qui ont été proposées. Le niveau le plus haut embrasse l'entièreté du cycle de conception d'une application interactive : software, hardware, fonctionnalités et interface. Le niveau inférieur ne s'attache qu'à l'interface homme-machine.

I.5.2.1. Méthodologies de conception d'une application interactive

Les méthodologies qui sont proposées pour l'élaboration d'une application interactive peuvent se décomposer en deux tendances distinctes : la première, et la plus ancienne, consiste à partir des fonctionnalités pour ensuite réaliser l'interface-utilisateur; cette dernière n'est en ce sens qu'un module supplémentaire d'entrée/sortie. L'interface-utilisateur est greffée aux fonctionnalités. La deuxième tendance est plus récente et centre la conception d'une application interactive sur l'utilisateur. Les fonctionnalités de l'application interactive sont à présent des services rendus à l'utilisateur. Elles sont appelées par ce dernier via l'interface. Nous présentons ci-dessous deux méthodologies utilisant la deuxième approche, à savoir la création d'une application interactive présentant à l'utilisateur un répertoire de fonctions que l'application peut réaliser. La méthodologie que nous présenterons au chapitre IV suit la permière tendance.

* Rubinstein et Hersh (1984) décrivent le processus de conception d'une application en 5 étapes :

- la collection d'informations, y compris les contraintes du marché, les spécifications techniques, l'état de l'art courant, les produits concurrentiels existants, les standard industriels... L'élément-clé est l'analyse de la tâche c'est-à-dire des activités réalisées par les utilisateurs dans leur environnement ;

- la conception de l'application interactive, y compris la spécification du mythe externe (la présentation offerte à l'utilisateur) et le modèle d'utilisation décrivant comment le système sera utilisé et comment il s'intégrera à la vie de l'utilisateur. L'application interactive consistera en un répertoire de fonctions et une interface qui sera la présentation externe des capacités du système interactif;

- l'implémentation et la construction d'abord d'un prototype, ensuite de l'application interactive en elle-même;

- le test et l'évaluation à la fois formelle et informelle du système;

- remise de l'application et évaluation des réactions des utilisateurs et du marché.

Le processus est itératif : les diverses étapes peuvent se répéter plusieurs fois. Les détails se trouvent dans [Rubinstein et Hersh, 1984].

* Schneiderman (1983) présente un cycle de vie d'un projet informatique quelque peu plus élaboré. Il est basé sur le besoin d'éducation et d'implication de la communauté des utilisateurs. Il comprend 8 étapes :

- initier le projet et collecter les informations,
- concevoir les structures sémantiques,
- concevoir les structures syntaxiques,
- spécifier l'équipement physique,
- développer le software,
- intégrer le système et le présenter aux utilisateurs,
- éduquer la communauté des utilisateurs,
- préparer un plan d'évolution.

Le plan d'évolution comprendra certainement la répétition de certains aspects des 8 phases. Tandis que Schneiderman ne précise pas si les fonctionnalités de l'application seront utilisées[9] par l'interface, sa démarche est avant tout centrée sur l'utilisateur. Par exemple, les diagrammes de flux ("task flow") qui seront conçus lors de la phase d'élaboration des structures sémantiques seront basés sur celui-ci. L'entièreté du processus de conception de l'application interactive se fera sur base du ou des utilisateurs futurs.

D'autres méthodologies sont exposées dans [Wasserman, Pircher & Shewmake, 1986; Buxton & Shniderman, 1980].

[9] Les fonctions deviennent alors des services rendus.

I.5.2.2. Méthodologie de conception de l'interface-utilisateur

Certaines méthodologies s'adressent plus particulièrement à la conception de l'interface-utilisateur. Nous présentons celle de Newman et Sproull (1979). Leur procédure commence par l'analyse de la tâche. Ensuite, la spécification de l'interface-utilisateur se décompose en 4 parties :

- **le modèle de l'utilisateur** : le modèle conceptuel qu'il se fait des informations qu'il voit et des actions qu'il doit réaliser. Ce modèle doit être le plus proche possible de la réalité. Dans l'exemple que nous développerons à partir du chapitre V[10], un bon modèle sera la représentation à l'écran du bon de commande que l'utilisateur enregistre. L'effort de compréhension des fonctionnalités du système et des opérations qu'il pourra effectuer sera ainsi fortement diminué.

- **le langage de commande** : après avoir compris le modèle conceptuel, l'utilisateur doit avoir à sa disposition des commandes pour le manipuler. Le langage qui lui sera offert doit être le plus naturel possible. Pour notre exemple, le click souris dans la zone d'introduction du nom du client sera la traduction du déplacement du stylo de l'utilisateur sur son bon de commande. Il est important de fournir un langage de commande qui soit à ce point naturel qu'on n'ait pas l'impression de l'employer. L'utilisation d'une calculette ne demande pas d'effort de la part de son utilisateur car son langage de commande rencontre le mode de pensée de ce dernier[11].

- **le feedback** : l'assistance de l'ordinateur pour la réalisation de la tâche; elle peut prendre la forme de messages d'aide, d'erreur... Elle représente ce que nous appelerons les messages interactifs (voir chapitre III et suivants). Dans notre exemple de saisie d'un bon de commande, un feedback du système interactif sera la présentation du libellé du produit dont on a introduit le numéro.

- **l'affichage d'information** : la présentation de l'état des informations que l'utilisateur manipule, en accord avec le modèle choisi. Le fait de présenter l'application interactive sous forme d'un bon de commande doit entraîner le traitement des informations qu'elle contient en accord le modèle conceptuel. Il serait par exemple stupide d'afficher le total de la commande autre part que dans le cadre réservé à cet effet.

1.5.3. Conclusion sur les principes et méthodologies

Malgré la diversité des approches, 3 grands thèmes semblent émerger parmi les auteurs. Il s'agit de l'**analyse de la tâche**, la **connaissance de l'utilisateur** et l'**approche itérative.**

L'analyse de la tâche n'est pas spécifique à la conception d'une application interactive. Il est clair qu'une application aussi conviviale et adaptée à l'utilisateur qu'elle soit, doit exécuter correctement les fonctions pour lesquelles elle a été développée. Ces dernières doivent également correspondre à la tâche effectuée par l'utilisateur[12].

---

[10] Il est présenté à l'annexe 1.

[11] Personne n'a l'impression d'utiliser un langage de commande lorsqu'il tape sur sa calculette 3 + 5 =...

[12] Une application interactive réalisant correctement le calcul de toute intégrale est de peu d'utilité si l'on désirait le calcul des racines carrées!

La connaissance de l'utilisateur est le but de la psychologie cognitive dont nous avons discuté à la section I.3.3. Rappelons que la diversité des utilisateurs est à la base des nombreux problèmes qui se posent dans la conception d'une application interactive puisqu'elle se doit de lui être adaptée.

L'approche itérative est la résultante de l'état-de-l'art en la matière. Nous sommes à l'heure actuelle incapables de réaliser une application interactive qui rencontrerait dès la première ébauche les objectifs qui ont été posés. L'itération est indispensable et un objectif qui en découle est la conception rapide d'un prototype pour permettre la réalisation des objectifs dans le laps de temps le plus court. Des outils permettent cette rapidité d'itération; nous les étudierons au chapitre suivant.

Ces 3 thèmes ne nous disent cependant pas comment réaliser un bon design d'application interactive. Ils nous donnent simplement des axes de réflexion pour y parvenir. La raison en est peut-être que la conception d'une application interactive est plus qu'une science, c'est un art.

I. 6. Conclusion

L'objectif de ce chapitre était de présenter au lecteur certains des problèmes qui pouvaient se poser au concepteur d'une application interactive. La solution miracle n'existe pas et une approche pluridisciplinaire est nécessaire pour prendre en compte les diverses considérations qui influencent la réalisation d'une application interactive.

Nous avons fait ressortir 3 principes importants dans la réalisation d'une application interactive : l'analyse de la tâche, la connaissance de l'utilisateur et l'itération du processus. Nous allons à présent étudier les outils qui sont à la disposition du concepteur pour faciliter son travail et surtout lui permettre d'atteindre plus rapidement une version exécutable de son application (et de la sorte réitérer plus rapidement le processus). C'est l'objectif du chapitre suivant.

Chapitre II
Techniques et outils de conception

II.0. Introduction

Le chapitre précédent nous a permis de montrer quelques uns des nombreux problèmes qui se posaient au concepteur d'une application interactive. De surcroît, il est confronté, d'un côté, aux desiderata des utilisateurs et, de l'autre, à l'inadéquation des outils qui lui sont généralement proposés pour les satisfaire. Ce chapitre procède à l'analyse de ces derniers.

Nous introduirons tout d'abord les qualités que l'on peut attendre d'outils de conception et de programmation avant de tenter de les classifier selon leurs caractéristiques. Nous analyserons ensuite plus en détails l'outil le plus prometteur en la matière : le Système de Gestion du Dialogue (SGD). Nous introduirons une taxonomie des SGD existants. Nous terminerons par quelques exemples d'outils actuels dans le domaine.

II.1. Qualités attendues d'un outil de conception et de programmation

Les qualités attendues d'un outil de conception et de programmation peuvent être générales ou plus spécifiques à un style d'interaction particulier.

II.1.1. Critères généraux

* Maclean et al (1986) ont identifié un nombre de critères que devait rencontrer un bon outil de conception et d'implémentation d'une application interactive :

- **Itération rapide autour du cycle de conception du dialogue** : L'approche itérative est obligatoire dans la conception d'une application interactive si l'on veut qu'elle rencontre les objectifs qui ont été fixés[1]. Il est alors souhaitable de disposer d'un outil qui permette la modification interactive du dialogue pour éviter que le concepteur ne doive, à chaque itération, éditer puis compiler l'entièreté du dialogue.

- **Spécification simple du dialogue** : Un outil de spécification doit permettre une manipulation aisée de la structure et du contenu du dialogue. Un outil trop complexe à manipuler est rapidement laissé de côté par son utilisateur. La spécification simple du dialogue facilite sa création et sa modification.

- **Gestion de plusieurs interfaces pour une même application :** La diversité des utilisateurs (cfr section I.3.1.) impose au concepteur de créer plusieurs interfaces pour une même application interactive. Un outil de conception doit permettre l'accès à une application via un ensemble d'interfaces-utilisateur.

- **Accès immédiat à une version fonctionnelle de l'application interactive** : La simulation de l'application interactive ou son exécution à une vitesse relativement lente ne

[1] Cfr section I.5.3.

permettent pas une évaluation et une comparaison efficaces des interfaces réalisées. Idéalement, la spécification du dialogue de l'application interactive devrait engendrer son implémentation sans qu'aucune compilation ne soit réalisée ultérieurement.

- **Minimum de contraintes sur les formes de présentation possibles** : L'outil ne doit pas imposer un certain style de conception, limitant les possibilités du concepteur dans la réalisation de son application interactive.

II.1.2. Critères spécifiques

* Henderson (1987) formula les caractéristiques qui lui semblaient souhaitables dans le cas précis d'outil de conception d'interfaces représentées par un cadre visuel tel qu'une fenêtre, un formulaire, une boîte de dialogue... Il s'agissait plus particulièrement du design de "control panels" (tableau de contrôle) tels qu'ils apparaissent lors de l'impression d'un document sur une machine similaire au Macintosh.

Nous reprenons ces caractéristiques, parfois judicieuses en dehors de leur contexte :

A partir du contexte de conception de l'outil :

- **La conception ne doit pas nécessiter de programmation** : Pour le type d'interface qui est ici considéré ("control panels"), l'outil doit permettre une conception interactive de l'interface.Le concepteur créera et définira les caractéristiques des objets de l'interface en les manipulant. La conception interactive permet au concepteur de voir immédiatement le résultat de ses actions et, de cette façon, de tester et évaluer rapidement l' interface[2].

- **La conception doit être rapide** : Ceci favorise le processus d'itération dans la conception de l'interface-utilisateur. Cela permet également de maintenir l'attention du concepteur éveillée et d'améliorer ainsi la profondeur d'exploration des différents problèmes auxquels il est confronté.

- **L'environnement doit permettre, en plus de la présentation, la conception du comportement de l'interface** : Une interface est une unité active; elle contient une dynamique qui doit elle-aussi être spécifiée et implémentée. L'outil qui sera offert devra prendre en compte cette dynamique, en plus de la statique de l'interface (sa présentation).

De par la nature de l'interface :

- **L'outil doit permettre des vues multiples de l'interface, reflétant son état** : Il est souhaitable d'offrir à l'utilisateur plusieurs présentations d'un même objet pour reflèter le contexte dans lequel il se trouve. Par exemple, la sélection d'un objet entraînera son affichage en mode vidéo inversé. De même, il peut être désirable de disposer de vues hiérarchiques d'un même objet. Une lettre d'un client pourra par exemple apparaître sous forme d'une icône, de texte, ou encore dans l'enveloppe de son expéditeur. L'utilisateur la manipulera tout comme il

---

[2] Hypercard , dont nous servirons pour implémenter l'exemple d'application interactive présenté à l'annexe 1 et au chapitre V, utilise cette approche.

le ferait d'une véritable lettre[3]. L'outil devra permettre de concevoir ces différents types de présentation.

- **L'outil doit permettre le passage d'un niveau de détail à un autre** : L'existence de vues multiples d'un même objet nécessite la possibilité de voyager parmi ces différentes vues.

- **La conception doit être basée sur une indépendance entre l'abstraction et la présentation** : Nous avons relevé à la section I.5.1. les bénéfices que cette séparation pouvait apporter.

- **Le concepteur doit pouvoir exprimer des interdépendances, interdisant certaines actions selon le contexte ou répercutant un changement sur les éléments impliqués** : Certaines actions de l'utilisateur peuvent être illégales; certains changements peuvent en provoquer d'autres. L'outil doit permettre de les spécifier, de même que les actions qui devront être entreprises une fois qu'ils ont été détectés.

A partir du processus de conception :

- **L'outil doit reconnaître la similarité entre les parties de l'interface** : Une interface sera souvent composée d'objets que nous qualifierons dans la suite de l'exposé de standard; un objet standard est un objet offert par l'environnement utilisé et que l'on peut façonner selon ses désirs en lui donnant les caractéristiques appropriées. L'outil doit permettre d'utiliser ce type d'objet en donnant également la possibilité au concepteur de définir ses propres types.

- **L'outil doit permettre des spécifications incomplètes** : Le concepteur doit pouvoir créer une occurence d'un type d'objet sans avoir à spécifier toutes ses caractéristiques. Chaque caractéristique contiendra lors de la création de l'objet une valeur par défaut que le concepteur pourra modifier après une analyse de la présentation et du comportement de l'objet pour qu'il réponde au mieux aux objectifs pour lesquels il a été créé.

- **L'outil doit permettre le partage de parties communes entre plusieurs interfaces** : Certaines parties d'interface sont communes à différentes applications interactives. On peut alors souhaiter une librairie d'éléments d'interface que se constituerait le concepteur. Il pourrait alors puiser dans celle-ci pour accélérer le processus de création de l'interface.

II.1.3. Conclusion

Les qualités que l'on peut attendre d'un outil de conception et d'implémentation d'une application interactive peuvent être générales ou plus spécifiques au style d'interface qu'elles permettront de réaliser.

Nous avons relevé au chapitre I les problèmes auxquels était confronté le concepteur d'une application interactive. L'outil de conception et d'implémentation est une application interactive au même titre que les applications qu'il permettra de développer. Le concepteur

[3] Il pourra ouvrir l'enveloppe, lire la lettre, la ranger...

devient l'utilisateur de l'outil de conception. Il faudra en conséquence lui offrir tous les éléments susceptibles de lui faciliter sa tâche de conception.

Nous analyserons à la section II.3. des éléments d'évaluation des outils de conception. Nous allons au préalable les classifier.

II.2. Classification des outils

Les outils pour le développement d'applications interactives peuvent être décomposés en 2 catégories : les boîtes à outils (toolbox) et les systèmes génériques. La présentation qui suit s'inspire largement des travaux réalisés par J. Coutaz dans [Coutaz 1986a, 1986b].

II.2.1. La boîte à outils

Un toolbox est une librairie de procédures pour l'écriture d'interface homme-machine. Ces procédures se font à différents niveaux d'abstraction. J. Coutaz en relève deux : le niveau de la gestion du poste de travail, grâce auquel le fonctionnement de l'appareil physique ne doit pas être connu, et le niveau de gestion du dialogue, au moyen d'entités simples (icônes, boutons, menus...) ou composées (formulaire ou boîte de dialogue).

Un toolbox a le mérite d'être extensible (le concepteur peut créer lui-même de nouvelles procédures) et flexible (il peut utiliser des procédures de bas niveau et contrôler finement l'interface qu'il réalise ou de plus haut niveau en laissant au toolbox le soin de gérer une grande part du contrôle de l'interaction avec l'utilisateur). Cependant, cette flexibilité a ses contreparties :

- *Il est d'un apprentissage difficile* : il faut connaître les spécifications de toutes les fonctions qui sont offertes,

- *Il n'aide pas le concepteur dans sa décomposition modulaire* : il ne force pas la séparation modulaire entre les fonctionnalités et l'interface de l'application interactive,

- *Il entraîne une duplication d'effort* : nous avons vu l'importance de l'itération dans la réalisation d'une application interactive pour rencontrer les besoins des utilisateurs. Or, on constate qu'une partie importante du code d'une interface est commune à toute application interactive. Ceci suggère la réutilisation de la partie commune. Les systèmes génériques permettent cette réutilisation.

II.2.2. Les systèmes génériques

Les systèmes génériques fournissent un squelette standard sur lequel sont greffés les composants spécifiques aux applications. Squelette et composants s'appuient sur les services d'un toolbox. On distingue 2 types de système générique : les Applications Extensibles et les Systèmes de Gestion du Dialogue.

II.2.2.1. Application extensible

Une application extensible regroupe le code commun à toute application interactive. Elle est constituée d'un squelette réutilisable qui doit être étendu pour répondre aux spécifications de l'application à concevoir.

Elle diminue fortement le temps nécessaire à la conception et l'implémentation de l'application. De surcroît, elle permet une uniformité des applications qui seront réalisées puisqu'elles partageront le même squelette d'application.

Le grand désavantage de cet outil est la nécéssité de la part du concepteur de connaître les fonctions du toolbox nécessaires à l'extension ou la modification du squelette de base. Il serait souhaitable de lui offrir un langage de haut niveau pour la spécification de son interface. C'est ce que proposent les Sytèmes de Gestion du Dialogue.

II.2.2.2.Système de Gestion du Dialogue (SGD)

Un SGD est un ensemble d'outils pour le traitement et l'administration de l'interface-utilisateur, un peu de la même manière qu'un Système de Gestion de Base de Données gère une Base de Donnée. Il est composé de deux modules : un préprocesseur, qui reçoit en entrée les spécifications de l'interface et génère son implémentation, et le noyau d'exécution, qui arbitre les interactions entre l'utilisateur et l'application.

"*La spécification est généralement une description textuelle exprimée dans un langage spécialisé [Hayes, 1983] ou dans une notation, extension de formalismes existants tels que les réseaux de Petri [Barthet, 1986], les grammaires BNF [Blesser, 1982; Olsen, 1983] ou les automates d'états finis [Wasserman, 1986]. Parfois, la spécification est effectuée de manière interactive comme dans Flair [Wong, 1982], Menulay [Buxton, 1983], SOS [Hullot, 1986] ou Peridot [Myers & Buxton, 1986]*" [Coutaz, 1986a].

La spécification interactive permet au concepteur d'une interface homme-machine de voir immédiatement l'effet de ses directives; elle permet d'atteindre plus rapidement le résultat recherché. De plus, elle permet dans certains cas la conception de l'interface sans aucune programmation[4].

Notons cependant que les SGD disponibles sont souvent spécifiques à un type particulier d'interface[5]; par exemple, RAPID/USE[6] est dédié à la création d'applications interactives d'interrogation de base de données [Maclean et al, 1986].

---

4 Voir section II.4.4.

5 Cfr section II. 1.

6 Voir section II.4.2.

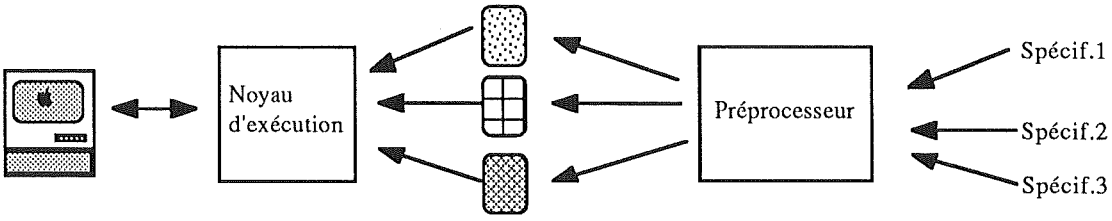


Figure II.1. : Schéma d'un Système de Gestion de Dialogue

II.2.2.3. Résumé

Les toolboxes mettent à la disposition du concepteur d'applications interactives un ensemble de procédures standard tandis que les systèmes génériques lui fournissent une architecture standard qu'il n'a plus qu'à compléter. Les applications extensibles proposent une interface de programmation en langage de programmation usuel tandis que les SGD permettent la spécification de l'application interactive à l'aide d'un langage de haut niveau ou, mieux encore, un langage graphique dans le cas de la spécification directe interactive.

Nous allons à présent analyser les critères à l'aide desquels on peut comparer les SGD existants. En effet, force est de constater qu'il existe de grandes différences dans leur conception et leurs caractéristiques.

II.3. Taxonomie des SGD

J. Coutaz reprend 3 critères introduits par Hayes (1985) et en propose 4 autres :

II.3.1. Niveau d'abstraction

Le niveau d'abstraction définit l'unité d'échange entre les fonctionnalités de l'application interactive et le SGD.

Un bas niveau d'abstraction sera par exemple la communication par le SGD aux fonctionnalités d'un évènement correspondant au click souris de l'utilisateur sur un objet particulier.

On parlera de haut niveau d'abstraction si, par exemple, le SGD interprète cet évènement, conclut qu'il s'agit d'un click souris sur un bouton signifiant la clôture de la saisie d'un message et communique alors aux fonctionnalités le message qui a été saisi par l'utilisateur.

L'exemple illustre bien qu'un bas niveau d'abstraction obligera le concepteur à un travail important de mise à niveau du dialogue avec les entités manipulées par les fonctionnalités. Il devra analyser les informations provenant du SGD pour fournir leur équivalent sémantique aux fonctions de l'application interactive, tâche qui devrait être réalisée par le SGD.

L'architecture que nous présenterons au chapitre suivant force un haut niveau d'abstraction car la communication entre l'interface et les fonctions de l'application se fera par l'échange de messages fonctionnels[7]. Par exemple, l'introduction du nom d'un client sera entièrement gérée par l'interface qui l'enverra lorsque l'utilisateur a terminé de le saisir.

II.3.2. Localisation du contrôle

"*Le contrôle du dialogue peut être interne, externe ou mixte [Tanner & Buxton, 1983]. Il est interne losqu'il réside dans les fonctions de l'application interactive. Il est externe lorsqu'il est maintenu par le SGD. Il est mixte lorsqu'il est géré alternativement par l'application et le SGD.*" [Coutaz, 1986a].

Le contrôle interne du dialogue oblige l'utilisateur à répondre aux demandes des fonctionnalités de l'application interactive sans qu'il lui soit possible de choisir l'ordre de marche qui lui conviendrait le mieux. Pour notre exemple, l'utilisateur se verrait contraint d'introduire en premier lieu le numéro du client alors qu'il pourrait souhaiter commencer par les lignes de la commande. Cette approche est inacceptable car nous avons vu au chapitre I que la conception d'une application interactive devait se faire en fonction de l'utilisateur. C'est lui qui doit diriger l'application interactive et ainsi être libre de choisir les actions qu'il désire effectuer. Le contrôle externe permet à l'utilisateur de prendre la direction des opérations.

Lorsque le contrôle est externe, les fonctionnalités de l'application interactive ne sont que des services rendus à l'utilisateur. Deux problèmes peuvent cependant se poser pour ce type de contrôle :

- l'occurence d'événements asynchrones par rapport aux actions de l'utilisateur. Les fonctionnalités de l'application interactive pourraient devoir communiquer à l'utilisateur des informations dont il n'a pas connaissance, comme par exemple l'arrivée d'un message provenant d'un autre utilisateur par le réseau de communication.

- le déclenchement d'une fonction peut conduire à un blocage de celle-ci car elle n'a pas à sa disposition toutes les informations nécessaires à son bon déroulement. Par exemple, si l'utilisateur juge qu'il a terminé la saisie de sa commande alors qu'il n'a pas fourni l'adresse du nouveau client, les fonctionnalités se verront dans l'imposibilité de traiter les informations fournies et resteront bloquées.

Le contrôle mixte permet de résoudre tous ces problèmes : l'utilisateur possède le contrôle de l'application interactive mais les fonctionnalités ont la possibilité de modifier son déroulement, par exemple pour signaler l'occurence d'un évènement ou demander des informations complémentaires.

---

[7] Un message fonctionnel contient des données sémantiques. Sa définition est donnée à la section III.2.1.

II.3.3. Ordonnancement des événements

L'ordre d'occurence des événements utilisateurs peut être spécifié explicitement, implicitement ou pas du tout.

L'ordonnancement explicite oblige le concepteur à spécifier toutes les actions que l'utilisateur pourra effectuer dans un contexte donné. Par exemple, l'apparition d'un menu à l'écran engendrera la spécification de toutes les actions possibles pour sélectionner un élément du menu. Ceci peut devenir fastidieux : le concepteur d'une application interactive utilisant la manipulation directe verra son travail décuplé s'il doit spécifier à tout moment les actions que pourra effectuer l'utilisateur à l'aide d'une souris. Les outils de spécification qui utilisent la technique des automates tels que USE, imposent au concepteur une spécification explicite.

L'ordonnancement implicite évite au concepteur de spécifier explicitement l'enchaînement des événement. Cet enchaînement peut être retiré des informations décrivant le comportement de l'interface. Son analyse permettra à l'outil de construire l'enchaînement qui devra être respecté[8].

Une approche intéressante pour dégager le concepteur des soucis que peuvent provoquer l'ordonnancement des événements est de combiner le contrôle mixte à un haut niveau d'abstraction. De la sorte, le concepteur ne devra ordonnancer que les événements significatifs pour les fonctionnalités; il ne sera pas concerné par l'enchaînement des événements de bas niveau tels que les clicks souris ou les touches au clavier. Nous utiliserons cette approche dans la méthodologie que nous définirons au chapitre IV; le concepteur décrira l'enchaînement des messages qui doivent être traités par l'utilisateur sans se soucier des actions que ce dernier effectuera [9].

II.3.4. Adaptabilité

L'adaptabilité est la faculté d'ajustement à la nouveauté d'une situation ou aux variations de l'environnement. Dans le domaine de l'interaction homme-machine, il s'agit de la capacité que possède l'interface de s'adapter à l'utilisateur [Diaper, 1986; Tardelli & Cooper, 1986].

Nous avons vu l'importance qu'il fallait accorder à l'étude de l'utilisateur pour concevoir une interface qui lui soit adaptée[10]; on peut concevoir deux raisons pour qu'elle soit non seulement adaptée, mais adaptable :

- L'application interactive est construite par itérations successives de manière à correspondre exactement aux modes de pensée de son utilisateur. L'interface est alors - dans le meilleur des cas car les connaissances ne sont pas à l'heure actuelle suffissantes pour réellement y parvenir - adaptée, mais elle doit encore être adaptable car la connaissance de l'utilisateur évoluera avec l'usage de l'application interactive. Au début, il sera peut-être un novice qui demandera beaucoup d'aide de la part de l'interface et se laissera dirigé. Au fur et à mesure qu'il progressera, il sera nécessaire

[8] Peridot, que nous décrivons à la section II.4.4. procède de la sorte.

[9] Voir section IV.1.4.

[10] Cfr section I.3.1.

de lui offrir une plus grande liberté dans l'accomplissement de sa tâche, des moyens d'optimiser son travail, par exemple à l'aide de "macros" exécutant en une fois un ensemble d'opérations qu'il effectue toujours dans le même ordre.

- un deuxième motif pour concevoir des interfaces adaptables est qu'une application interactive sera généralement conçue pour une classe d'utilisateurs qui, s'ils partagent certaines caractéristiques, n'en sont pas moins différents. Il faut alors leur offrir un moyen de personnaliser leur interface pour qu'elle réponde au mieux à leurs besoins. Dans notre exemple de saisie d'une commande, on offrira la possibilité de modifier les messages présentés par l'interface[11].

J. Coutaz fait la distinction entre une interface adaptable et adaptative. Une interface adaptable nécessite de la part de l'utilisateur une intervention explicite; il la personnalisera en effectuant les actions qui lui sont offertes pour y parvenir. Par contre, une interface adaptative analyse les actions de l'utilisateur et en déduit un profil sur lequel elle basera son comportement. Ceci est malheureusement du domaine de la recherche. L'intelligence artificielle se penche depuis un certain nombre d'années sur la question. On parle alors d'interface "intelligente" [Rich, 1983; Diaper, 1986].

II.3.5. Parallélisme

Le parallélisme revêt 3 formes complémentaires dans le domaine de l'interaction homme-machine :

- l'utilisation simultanée de plusieurs unités physiques d'entrée-sortie;
- l'interaction simultanée de l'utilisateur avec plusieurs applications ou plusieurs objets d'une application (l'utilisateur peut introduire en parallèle le numéro du client et les lignes de produit de la commande);
- la communication simultanée entre plusieurs utilisateurs.

Les deux premiers types de parallélisme sont disponibles sur les stations de travail offrant le "multi-tâches". Cependant, ils sont encore peu adaptés au cadre de l'interaction homme-machine.

La communication simultanée entre plusieurs utilisateurs introduit une dimension sociale à l'étude des applications interactives. Elle n'a pas encore fait l'objet de réflexions déterminantes.

II.3.6. Généralité

Nous avons débuté ce chapitre par l'analyse des qualités que l'on pouvait a priori attendre d'un outil de conception d'une application interactive. Nous avions distingué les qualités générales des qualités spécifiques à un style d'interaction. Un critère de classification des SGD existants concerne le niveau de généralité dans les applications interactives qu'ils permettent de créer. Il existe en effet une grande variété de types d'application interactive; certaine nécessitent par exemple un contrôle fin des appareils d'entrée-sortie comme c'est le cas

[11] Voir sections IV.1.4.2. et VI.4.1.5.

pour les applications graphiques, tandis que d'autres ne nécessitent pas d'appareils sophistiqués pour dialoguer avec l'utilisateur. Les styles d'interaction qui pourront être utilisés constituent également un élément d'évaluation de la généralité de l'outil.

Un haut niveau de généralité permettra de concevoir un nombre important d'applications interactives de types et/ou de styles différents. Par exemple, il sera possible à partir des spécifications de la phase de saisie d'une commande d'implémenter l'application interactive aussi bien sur un environnement de type traditionnel utilisant un langage de commande qu'un environnement de manipulation directe avec une gestion graphique de haut niveau.

En résumé, un haut niveau de généralité est souhaitable car il offrira au concepteur d'application interactive la possibilité de spécifier le style d'interface qui conviendra le mieux au type d'application qu'il souhaite réaliser.

II.3.7. Contexte

"*Un contexte est une structure d'éléments déterminant le sens d'une situation*" [Coutaz, 1986a]. Le contexte d'une application interactive doit permettre à l'utilisateur de se situer par rapport à l'accomplissement de sa tâche. Un travail considérable en ce domaine a été réalisé par Nivergelt (1985) en proposant le modèle Sites, Modes and Trails. Il a remarqué que les questions que se posent généralement les utilisateurs non familiers avec une application interactive particulière sont du type:

- Où suis-je?
- Que puis-je faire?
- Comment suis-je arrivé ici?
- Où puis-je aller et comment?

Une application interactive doit permettre à l'utilisateur de répondre à ces questions à tout moment. Trois concepts sont proposés par Nievergelt pour permettre d'y répondre :

* Site : l'environnement des données courantes, accessibles à un moment donné;
* Mode : l'ensemble des commandes couramment utilisables;
* Trail : l'historique du dialogue (les données historiques et les commandes qui ont été effectuées).

La richesse du contexte détermine l'effet et la qualité des services qui lui sont étroitement liés tels que l'aide [Bösser, 1987)], l'historique et la reprise sur erreur. Il s'agit d'offrir à l'utilisateur un véritable tableau de bord à partir duquel il pourra obtenir toutes les informations qui pourraient l'aider dans la réalisation de sa tâche. C'est une aide généralisée qui, au même titre que l'application interactive, doit être adaptée aux spécificités de l'utlisateur. Un chevronné préferera ne pas être gêné dans sa tâche par l'occurence de messsages intempestifs lui signalant des informations dont il a connaissance alors qu'un débutant prendra soin de toutes les informations qu'on lui communiquera. La richesse des informations à fournir dépendra donc des caractéristiques de l'utilisateur. Une bonne gestion du contexte pour notre exemple serait de fournir à l'utilisateur le nom et l'adresse du client dès qu'il a introduit son numéro, de lui

afficher le libellé du produit dont il a saisi le code... Nous verrons que le type d'interface aura une influence sur la qualité du contexte que l'on désirera fournir[12].

II.3.8. Conclusion

A partir des éléments repris ci-dessus, on peut considérer qu'un bon Système de Gestion de Dialogue devrait :

- présenter un haut niveau d'abstraction,
- permettre un contrôle mixte,
- permettre de spécifier l'ordonnancement implicitement,
- pemettre la conception d'applications interactives adaptatives,
- permettre le parallélisme,
- être d'un haut niveau de généralité, et
- gérer le contexte de l'application.

Nous terminerons ce chapitre par quelques exemples d'outils de conception actuellement disponibles ou en cours de réalisation. Nous les comparerons aux critères que nous venons de définir.

II.4. Exemples d'outils

Cette section a pour objectif de résumer l'approche suivie dans la réalisation de quelques outils de conception d'applications interactives. L'objectif de ces outils est avant tout de parvenir le plus rapidement possible à un prototype, de manière à pouvoir réitérer le processus de conception un nombre de fois suffisant que pour obtenir des résultats satisfaisants.

II.4.1. MacApp

[Schmucker, 1986]

MacApp est une application extensible réalisée sur Macintosh selon une approche orientée objet. Elle fournit au concepteur d'applications interactives tous les standard d'interface du Macintosh tels que les menus, les fenêtres...C'est en quelque sorte une application interactive sans fonctionnalité : le concepteur doit définir le contenu des fenêtres et des menus du squelette. L'approche orientée objet offre des avantages qui seraient trop long à exposer ici et qui dépassent quelque peu le cadre de l'exposé[13]. L'auteur parle d'une réduction du temps de conception et du code à écrire d'un facteur de 4 à 5.

Niveau d'abstraction

Le concepteur devra lui-même implémenter la séparation modulaire entre les fonctionnalités et l'interface. Dans le cas où il l'effectue (ce qui n'est garanti nullement par l'outil), il pourra définir un niveau d'abstraction quelconque. Le code commun de l'interface lui évitera cependant de se concentrer sur les actions de bas niveaux de l'interface et favorisera la réalisation d'un haut niveau d'abstraction.

---

12 Voir section V.1.4.

13 Voir cependant Hypercard (annexe 3).

Localisation du contrôle

La localisation du contrôle n'est pas totalement fixée par MacApp. Le principe de tout environnement utilisant la manipulation directe (comme le Macintosh) est de réagir aux actions de l'utilisateur pour déclencher les traitements appropriés. On peut alors parler de contrôle externe. Celui-ci peut cependant devenir un contrôle mixte si le concepteur permet aux fonctionnalités d'interrompre les actions de l'utilisateur en fonction de ses besoins. La localisation sera donc externe ou mixte.

Ordonnancement des événements

MacApp constitue le code commun de toute application interactive; il ne permet pas sa spécification. La notion d'ordonnancement des événements n'a donc pas de signification pour un outil permettant l'implémentation et non la spécification d'un système interactif.

Adaptabilité

Le logiciel est divisé en deux parties : son code et ses ressources. Les ressources d'une application représentent les informations qu'elle manipule et qui sont enregistrées dans des fichiers éditables en dehors de l'application (les fichiers de ressources). L'utilisateur peut alors modifier ces informations[14] pour les adapter à ses besoins sans que l'application ne nécessite une recompilation. On peut alors parler d'interface adaptable. Remarquons que cette possibilité est offerte à toute application conçue sur Macintosh; ce n'est donc pas le propre de MacApp.

Parallélisme

Le menu, la fenêtre, l'icône et la manipulation directe sont les styles d'interaction utilisés par le Macintosh. Nous avons signalé à la section II.3.5. que la manipulation directe permettait de travailler parallèlement sur plusieurs objets. MacApp permettra donc au moins ce type de parallélisme (l'utilisation simultanée de plusieurs unités physiques et l'interaction avec plusieurs application seront également offert sur une station de travail "multi-tâches").

Généralité

Nous avons précisé ci-dessus les styles d'interaction offerts par le Macintosh. MacApp permettra la création d'applications utilisant ces styles. Le niveau de généralité sera par conséquent élevé puisqu'elles couvrent un large éventail des types d'application interactive.

Contexte

Il n'existe pas de gestion de contexte offert par MacApp. Le concepteur devra entièrement créer les mécanismes et les informations qu'il souhaite offrir à l'utilisateur pour l'aider dans la réalisation de sa tâche.

---

[14] Via un logiciel spécialisé d'édition.

II.4.2. RAPID/USE

[Wasserman, 1986]

Wasserman a conçu une méthodologie de spécification et d'implémentation d'application interactive : USE (User Software Methodology). La phase initiale d'analyse comprend, en plus de la modélisation traditionnelle des données et des traitements, l'identification des caractéristiques de l'utilisateur et de l'environnement dans lequel le système sera utilisé. La seconde étape est le design externe, incluant le design de l'interface. La troisième étape est la création d'une version exécutable de l'interface homme-machine pour permettre à l'utilisateur et au développeur d'analyser en commun le prototype et de décider des améliorations possibles. Il y a donc clairement itération entre ces étapes.

L'outil fourni par la méthodologie USE est le système RAPID/USE composé d'un interpréteur de diagrammes de transition : le TDI (Transition Diagram Interpreter) et l'Action Linker.

Les spécifications de l'interface sont définies sous forme de diagrammes de transition d'état. Il est possible d'obtenir une version exécutable à partir des spécifications en les transformant en tables par le TDI. L'Action Linker permet d'associer les fonctions de l'application à l'interface.

RAPID/USE est donc un outil qui permet à la fois de construire et de valider l'interface ainsi que de construire un système fonctionnel (associer les fonctions de l'application à son interface). Il tourne sur UNIX et est largement utilisé et documenté. Tandis qu'il utilise des techniques graphiques[15], il ne permet la conception que d'interfaces non graphiques (basées sur le texte).

Niveau d'abstraction

Le niveau d'abstraction est défini par le concepteur de l'application interactive; la spécification de l'interface se faisant à un bas niveau (la frappe des touches clavier), il y a une nécessité de mise à niveau de celle-ci (et non des fonctionnalités) pour obtenir un haut niveau d'abstraction. Ceci est tout-à-fait acceptable puisque cela permet au spécifieur de l'interface une grande précision sans perturber les fonctionnalités de leurs activités sémantiques.

Localisation du contrôle

Le contrôle est externe : l'utlisateur contrôle l'application interactive en déclenchant les fonctions appropriées. Il est toutefois possible au concepteur de parvenir à un contrôle mixte en spécifiant des paramètres de retour que l'interface interprétera comme une demande complémentaire ou l'occurence d'un événement asynchrone. Le mécanisme est cependant complexe et constitue plus de la "cuisine interne" qu'un véritable contrôle mixte.

[15] La création des diagrammes de transition peut se faire à l'aide d'un éditeur graphique.

Ordonnancement des événements

RAPID/USE utilise un automate d'états finis utilisant des spécifications sous forme de diagrammes de transition d'état. L'ordonnancement des événements est explicite : le concepteur doit spécifier explicitement l'enchaînement des événements que devra respecter son interface.

Adaptabilité

Aucun mécanisme n'est offert par l'outil pour concevoir une application interactive adaptable ou adaptative. C'est entièrement à la charge du concepteur.

Parallélisme

Le parallélisme pourrait être envisagé en laissant la possibilité à l'utilisateur de voyager dans plus d'un diagramme de transition à la fois. Cette possibilité n'est cependant pas implémentée et n'offre que peu d'avantage dans le contexte d'utilisation de cet outil (voir paragraphe suivant).

Généralité

L'outil a été spécialement conçu pour des applications interactives d'interrogation de base de données. Il pourrait néanmoins être utilisé pour la plupart des applications interactives utilisant un langage de commande. Par contre, son extension à des application utilisant la manipulation directe pose des problèmes de lourdeur dans la notation qui en résulterait.

Contexte

La spécification par diagramme de transition d'état permettrait la gestion du contexte telle que nous en avons parlé avec le modèle Site, Mode and Trails puisqu'il suffirait de communiquer à l'utilisateur sa position dans le graphe de transition d'état avec ce qu'il peut effectuer et un historique de ses dernières opérations. Cette possibilité n'est cependant pas exploitée par le SGD.

II.4.3. Omega

[Métais, 1986]

"*Omega est un logiciel d'aide à la définition, à la réalisation et à la maintenance d'applications à dominante transactionnelle. Il permet de gérer de façon plus ou moins automatique le dialogue avec l'utilisateur*" [Petoud, 1986]. Il s'appuie sur 3 modèles :

- le domaine des données (au niveau logique et physique)
- le domaine de la conversation
- le domaine de la présentation

Le domaine de la conversation s'appuie sur un scénario défini par le concepteur. L'approche est comparable à USE si ce n'est le langage de plus haut niveau qu'offre Omega par rapport aux diagrammes de transition. La conversation permet de dissocier dans le dialogue les aspects sémantiques des aspects syntaxiques ou lexicaux (la syntaxe concrète).

Le domaine de la présentation sert à préciser la forme et le contenu des écrans qui seront affichés pendant le déroulement de la conversation. Il constitue en quelque sorte la statique du dialogue tandis que la conversation traite de la dynamique.

Omega permet le prototypage des écrans, de la conversation et de la validation de la saisie. Il est alors possible de tester rapidement la spécification du dialogue sans aucune programmation traditionnelle.

Il génère automatiquement une part prépondérante du code à partir de la définition abstraite de l'application interactive.

Niveau d'abstraction

Le niveau d'abstraction offert par Omega est de haut niveau car il fait une séparation nette entre les aspects sémantiques et syntaxiques de l'interface. Les fonctions de l'application interactive communiqueront avec l'interface par l'échange d'informations sémantiques.

Localisation du contrôle

Le contrôle est mixte : l'utilisateur peut orienter le déroulement de l'application interactive en effectuant des choix parmi les menus qui lui sont offerts; les fonctionnalités peuvent à tous moment réorienter le déroulement de l'application par exemple lorsque l'utilisateur n'a pas introduit toutes les données nécessaires, lorsqu'il a commis une erreur...

Ordonnancement des événements

L'ordonnancement est explicite, au moyen d'un scénario que définira le concepteur. Ce scénario peut lui-même être décomposé en phases. A une phase est associé un enchaînement d'échanges, qui peut contenir des parties conditionnelles ou répétitives.

Adaptabilité

Tout comme RAPID/USE, Omega n'offre pas de mécanisme d'adaptation à l'utilisateur. Le concepteur devra les réaliser entièrement.

Parallélisme

Bien qu'il soit à nouveau possible d'utiliser la méthode de spécification pour décrire le parallélisme, l'outil ne permet pas son implémentation. En revanche, son utilisation sur une station de travail "multi-tâches" permettra l'exécution simultanée de plusieurs applications interactives ou l'utilisation en parallèle de plusieurs unités d'entrée-sortie.

Généralité

Oméga est un logiciel d'aide à la définition, la réalisation et la maintenance d'applications à dominante transactionnelle. Ceci peut être rapproché de ce qui a été dit pour RAPID/USE.

Contexte

Une faible gestion de contexte est proposée par Omega; s'il est possible d'afficher de l'aide, d'effectuer des retours en arrière, on constate à nouveau que certaines informations qui seraient utiles à l'utilisateur telles que sa position dans le scénario qui a été défini, ne lui sont pas accessibles.

II.4.4. Peridot (Programming by Example for Real-time Interface Design Obviating Typing)

[Myers & Buxton, 1986]

Peridot est un SGD pour la création d'applications interactives dans un environnement utilisant la manipulation directe. Le principe de Peridot est de laisser le concepteur lui démontrer comment il veut qu'apparaisse et fonctionne l'interface qu'il spécifie. Pour cela, le concepteur n'a pas de programmation à faire; il réalise son interface par Manipulation Directe en spécifiant à la fois la présentation et les actions que l'utilisateur pourra effectuer. On peut de la sorte imaginer que des non-programmeurs puissent concevoir et implémenter leurs interfaces sans rien devoir programmer.

Les buts de Peridot sont :

1. de pouvoir être utilisé pour tout style d'interaction compatible avec la Manipulation Directe;

2. que le système soit facile à utiliser par le concepteur et ne requiert que peu ou pas d'apprentissage;

3. que le concepteur ne doive pas avoir à programmer;

4. que l'interface soit visible à tout moment lors de son développement et que les changements soient immédiatement apparents;

5. que le comportement de l'interface puisse également être créé par manipulation directe et fonctionner en temps réel;
(les points 4 et 5 permettent un prototypage extrêmement rapide)

6. que le système puisse créer un code objet de l'interface suffisamment performant pour qu'il puisse être utilisé dans l'implémentation finale de l'application interactive.

Pour réaliser de telles performances, Peridot travaille par inférence. Le système essaie de deviner, d'inférer la relation que peut avoir l'action qu'effectue le concepteur avec les éléments existant dans l'interface et demande ensuite au concepteur si son inférence est exacte. Si ce n'est pas le cas, il essaie à nouveau de trouver l'inférence qui correspond à la volonté du concepteur. Notons que le concepteur a également la possibilité de spécifier explicitement cette correspondance de même qu'il lui est possible de programmer les actions à effectuer à l'aide du langage Lisp.

Trois types d'inférences peuvent être réalisées :

- inférence entre objets : lorqu'on dessine un objet, le système cherche avec quel ou quels autres objets il peut être relié. Par exemple, le concepteur dessine un rectangle en gris clair, puis dessine un rectangle noir légèrement décalé pour réaliser de la sorte un rectangle ombré. Le système lui demandera alors si les deux rectangles doivent être de même dimension, ce à quoi l'utilisateur répondra oui;

- inférence itérative : lorsque le concepteur affiche deux objets d'une liste, le système réagit en demandant s'il faut afficher toute la liste de la même manière. Si l'utilisateur lui répond affirmativement, il s'exécute et évite à l'utilisateur d'effectuer cette tâche, souvent fastidieuse;

- inférence conditionnelle : pour traiter les cas spéciaux ou d'exception, il est nécessaire d'utiliser une condition qui, lorsqu'elle est vraie, produit l'effet correspondant (par exemple dans une alternative, il faut mettre une croix dans l'élément qui a été choisi).

Ceci concernait la spécification de la présentation de l'interface. La spécification de son comportement se fait en alternance entre deux modes. Le concepteur peut se comporter en tant que spécifieur ou en tant qu'utilisateur. Lorqu'il se comporte en tant que spécifieur, il a à sa disposition des icônes représentant les appareils (tels que la souris) que pourra manipuler l'utilisateur. Il travaillera en mode utilisateur pour spécifier les actions qui doivent se passer en temps réel comme le double click ou l'animation.

Comme le concepteur voit à tout moment l'interface qu'il réalise, il pourra très rapidement la valider ou l'améliorer. La possibilité qu'il a de travailler en mode utilisateur accentue encore cette possibilité.

En résumé, Peridot est un outil très prometteur de conception d'interfaces utilisant la manipulation directe. Il serait intéressant de pouvoir analyser plus profondément ses capacités pour voir si elles sont effectivement comparables à ce qui est annoncé. L'implémentation de Péridot n'était pas encore terminée au moment de la rédaction du document sur lequel nous nous sommes basés.

Niveau d'abstraction

Peridot permet de construire des procédures qui gèreront l'interface de l'application interactive. Ce seront les fonctionnalités de l'application qui appeleront ces procédures pour qu'elles interagissent avec l'utilisateur. Le niveau d'abstraction sera alors celui que le concepteur choisira. Il est tout-à-fait possible de parvenir à un haut niveau d'abstraction car Peridot permet de spécifier l'interface selon une approche de bas niveau; la gestion de ces événements sera donc prise en compte par l'interface et les fonctionnalités n'en n'auront pas connaissance.

Localisation du contrôle

Le contrôle de l'application interactive peut être qualifié de mixte. Les fonctionnalités appelerons les procédures adéquates de l'interface (contrôle interne) mais à l'intérieur d'une procédure, l'utilisateur contrôlera le déroulement de l'application interactive (contrôle externe).

Ordonnancement des événements

L'ordonnancement est implicite; le concepteur démontrera les actions que peut réaliser l'utilisateur sans devoir spécifier leurs enchaînements. Celui-ci sera déterminé par Peridot en analysant les actions du concepteur.

Adaptabilité

Peridot a été conçu pour pouvoir être utilisé par des non programmeurs. On peut alors imaginer que l'utilisateur d'une application interactive pourra éditer l'interface qui lui est présentée de manière à ce qu'elle corresponde tout-à-fait à ses désirs. On peut dès lors parler d'interface adaptable. L'adaptativité n'est une nouvelle fois pas gérée automatiquement et doit être réalisée par le concepteur.

Parallélisme

L'interaction simultanée de l'utilisateur avec plusieurs objets de l'application est tout-à-fait réalisable dans un environnement de manipulation directe. De même, l'utilisation de Peridot sur une station de travail offrant le "multi-tâches" permet l'utilisation simultanée de plusieurs unités physiques d'entrée-sortie ou encore l'interaction avec plusieurs applications interactives.

Généralité

Peridot a été conçu uniquement pour la spécification d'interfaces utilisant la manipulation directe. Ce style d'interaction est cependant très vaste et offre au concepteur un large choix parmi les types d'application interactive qu'il pourra réaliser.

Contexte

La gestion du contexte est à charge du programmeur.

II.4.5. Conclusion sur les exemples d'outils

Un tableau récapitulatif est présenté à la figure II.2. L'exposé de ces quelques outils de conception et leur analyse par rapport aux critères que nous avons spécifiés met en lumière les points faibles de la plupart des outils actuels :

	MacApp	Use	Omega	Peridot
Abstraction	défini par le concepteur	défini par le concepteur	haut niveau	haut niveau
Contrôle	externe ou mixte	externe	mixte	mixte
Ordonnanc.	----------	explicite	explicite	implicite
Adaptabilité	semi-adaptable	à charge du concepteur	semi-adaptable	oui
Parallélisme	entre objets	non	possible	possible
Généralité	élevé	interrogation de BD	interrogation de BD	manipulation directe
Contexte	non	non	faible	non

Figure II.2. : Tableau récapitulatif des exemples d'outils

Ils offrent très peu d'informations à l'utilisateur sur le contexte dans lequel il se trouve. C'est d'autant plus regrettable que les informations dont il a besoin sont parfois utilisées par l'application interactive mais ne lui sont pas disponibles.

L'adaptation de l'interface à l'utilisateur est un autre problème, loin d'être résolu à l'heure actuelle. Si certaines applications interactives peuvent être qualifiées d'adaptables à l'utilisateur, il s'agit habituellement de changements superficiels. Peridot semble l'approche la plus intéressante en ce domaine. Quant à l'obtention d'une application interactive adaptative, nous n'en sommes encore qu'aux balbutiements. Il est clair cependant que l'adaptation automatique de l'interface à l'utilisateur est le but ultime de tout concepteur d' application interactive.

On constate que les deux éléments repris ci-dessus sont des lacunes que l'on peut reprocher au noyau d'exécution du SGD et non à son préprocesseur, chargé de traduire les spécifications de l'interface. Nous avons également signalé que le parallélisme était quelque fois possible mais une nouvelle fois en dehors du SGD. Ce critère est cependant lié avant tout au matériel qui sera utilisé.

Il n'est pas réellement possible de créer un SGD permettant de concevoir n'importe quel type d'application interactive [Tanner & Buxton, 1985]. Une approche tout-à-fait générale ne peut se faire à cause de la trop grande variété qui peut exister parmi celles-ci. Un SGD permettant de concevoir toute application interactive utilisant la manipulation directe ou encore n'importe quel langage de commande nous semble posséder un haut niveau de généralité.

Un point positif que l'on peut reconnaître est le haut niveau d'abstraction dû à l'acceptation commune d'une séparation nette entre les éléments sémantiques et ceux qui concernent la présentation; les fonctionnalités de l'application interactive ne manipulent que les informations sémantiques et l'interface traduit les actions de l'utilisateur en unités sémantiques, compréhensibles par les fonctionnalités.

II.5. Conclusion

Nous avons vu au chapitre I que la conception d'une application interactive était une tâche complexe : le concepteur se voit confronté à un ensemble de problèmes dont les solutions ne font pas toutes partie de la même discipline.

L'objectif de ce chapitre était de recenser les types d'outils offerts au concepteur pour l'aider dans la réalisation de sa tâche. Ces outils ne se situent pas tous au même niveau; certains se contentent d'aider le programmeur dans l'implémentation de son application tandis que d'autres lui permettent de spécifier l'interface dans un langage de haut niveau et gèrent eux mêmes le dialogue avec l'utilisateur.

Les efforts à réaliser en vue d'obtenir des outils puissants constituant une véritable aide au concepteur doivent encore être fournis. Nous avons souligné l'approche intéressante qui consistait à permettre au concepteur de spécifier interactivement son interface.

Nous en avons terminé avec le contexte dans lequel nous pouvons situer ce travail. Les deux chapitres qui suivent ont pour but de présenter au lecteur la méthodologie qui a été réalisée dans le cadre d'une extension de la méthode de spécification proposée par F. Bodart et Y. Pigneur dans [Bodart, 1983] en vue de permettre la spécification d'applications interactives.

Le chapitre suivant présentera le modèle d'application interactive sur lequel nous nous sommes basés.

Chapitre III
Modélisation et architecture d'une application interactive

III.0. Introduction

Le premier chapitre nous a permis d'introduire la problématique de la conception d'une application interactive. Nous avons insisté sur les nombreuses difficultés qui pouvaient survenir et qui nécessitaient une approche pluridisciplinaire.

Le deuxième chapitre a traité des outils qui existaient pour aider le concepteur dans la réalisation de son application interactive. Nous avons vu que de nombreux efforts restaient à faire en vue de bénéficier d'outils vraiment efficaces.

Ce chapitre est une introduction à la méthodologie que nous proposerons au chapitre 4. Il pose les fondements sur lesquels nous nous sommes appuyés pour la concevoir.

Une application interactive peut être décomposée en deux éléments principaux : les fonctionnalités et l'interface-utilisateur. Le dialogue qui a lieu entre l'application interactive et l'utilisateur peut être modélisé de manière à mettre en évidence d'un côté la tâche qui doit être réalisée et, de l'autre, la présentation qui lui est attachée. Cette modélisation sera à la base de notre méthodologie.

La section suivante pose les bases de l'architecture d'application interactive que nous affinerons tout au long de ce chapitre.

III.1. Séparation modulaire entre les fonctionnalités et l'interface

Le chapitre I nous a permis de définir ce que nous entendions par application interactive. Cette section a pour but de préciser l'architecture sur laquelle nous nous baserons pour notre méthodologie de spécification. Elle sera précisée au fur et à mesure que nous présenterons les éléments de modélisation du dialogue.

Un principe fondamental que nous avions introduit au chapitre I est celui de la séparation modulaire entre l'interface-utilisateur et les fonctionnalités de l'application interactive. Ce principe est à la base de notre architecture. Nous pouvons de la sorte décomposer une application interactive en 2 composantes principales :

- **l'interface homme-machine** est le composant de l'application interactive en contact avec l'utilisateur. Il est chargé de prendre en compte toutes les actions de l'utilisateur, de lui afficher les informations dont il a besoin. C'est à lui d'activer les fonctions qui correspondent aux actions de l'utilisateur et de lui afficher les résultats des traitements. Les actions qu'effectuera ce dernier n'auront pas toutes un impact sur les fonctionnalités de l'application interactive; l'interface homme-machine se chargera entièrement de ces actions. L'interface est donc l'outil de communication permettant à l'utilisateur de réaliser la tâche qu'il s'est fixée.

Une interface-utilisateur devra répondre aux besoins que nous avons détaillés aux chapitres I et II; elle devra rencontrer les désirs de l'utilisateur, être adaptée au modèle qu'il se fait de la tâche à effectuer, l'aider dans la réalisation de celle-ci en lui présentant toutes les informations dont il a besoin... La spécification de l'interface homme-machine devra permettre de définir explicitement tous ces éléments.

- **les fonctionnalités de l'application interactive** constituent la partie sémantique de l'application. Elles vont traiter les données fournies par l'utilisateur, en fournir d'autres, accéder à une base de données... Elles sont activées par l'interface qui a lui même reçu la demande correspondante de l'utilisateur, et lui fournissent les résultats des traitements effectués.

  Dans la suite de l'exposé, nous parlerons indifféremment de fonctionnalités et d'application proprement dite pour désigner ce composant.

L'application proprement dite n'est pas concernée par la présentation qui sera faite des éléments qu'elle manipule. Par exemple, un entier pourra tout aussi bien être représenté par une règle graduée qu'un camembert ou encore le chiffre correspondant. Les fonctionnalités sont indépendantes de cette représentation. Ceci permet la conception séparée des deux composantes. Dans un but de conception itérative qui est, nous l'avons vu, crucial en matière de développement d'application interactive, l'interface pourra être développée séparément des fonctionnalités et donc raffinée pour correspondre aux besoins , sans que l'application proprement dite ne doive être modifiée. Ceci est la raison primordiale de la séparation.

Le type d'application interactive que nous modéliserons s'appuiera généralement sur une base de données contenant toutes les informations manipulées par l'application proprement dite. La base de données constitue un troisième composant auquel nous ne nous attacherons cependant pas dans ce travail.

Ceci nous donne l'architecture d'application interactive suivante :

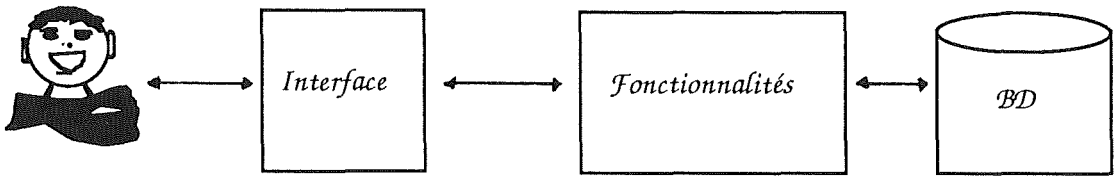


Figure III.1. : architecture d'une application interactive

Le but de ce travail est de proposer une méthodologie de spécification s'appuyant sur la méthodologie de spécifications fonctionnelles proposée par F. Bodart et Y. Pigneur (1983). Tous les éléments de spécification des fonctionnalités et de la base de données y sont largement explicités. Nous ne nous pencherons ici que sur une extension de la méthodologie en vue de spécifier le composant interface-utilisateur.

La section suivante présentera plus en détails la première composante de notre architecture : les fonctionnalités de l'application interactive.

III.2. Les fonctionnalités

Les fonctions manipulent les données sémantiques de l'application interactive. Elles effectuent les traitements pour lesquels elles ont été conçues et ne connaissent pas la présentation des entités qu'elles créent, modifient ou détruisent.

Le mécanisme qui permet aux fonctions de communiquer entre elles ou avec l'interface est l'envoi de messages fonctionnels :

III.2.1. Définition d'un message fonctionnel

Un message est "*une collection structurée d'informations véhiculées*" [Bodart & Pigneur, 1988]; il est constitué d'attributs d'un type défini et sur lesquels peuvent porter des contraintes d'intégrité. Un message est comparable à un record en Pascal ou à une structure en C sur lequel le spécificateur peut poser des contraintes. Une contrainte sera par exemple un intervalle de valeur pour une entier, la somme de deux attributs ne devant pas dépasser une limite...

Un message fonctionnel est un message à destination ou originaire des fonctions de l'application interactive. On distingue deux types de message fonctionnel [Chandelon & Warnant, 1987) : les messages fonctionnels **externes** sont ceux qui communiquent avec l'environnement. Ils nous intéressent particulièrement puisque l'agent communicateur de ces messages vers ou en provenance de l'environnement sera l'interface-utilisateur. Le deuxième type de message fonctionnel est constitué des messages fonctionnels **internes**; ils ne concernent que l'application proprement dite. Il s'agit de messages que s'échangent les traitements pour signaler des changements d'état de l'application. Dans la suite de l'exposé, nous désignerons par message fonctionnel les messages fonctionnels communiquant avec l'environnement c'est-à-dire les messages fonctionnels externes.

III.2.2. Résumé

Le moyen de communication entre les fonctionnalités de l'application interactive et son interface est l'utilisation de messages fonctionnels externes. Les fonctionnalités s'échangent entre elles des messages fonctionnels internes. Ceci est présenté à la figure III.2.

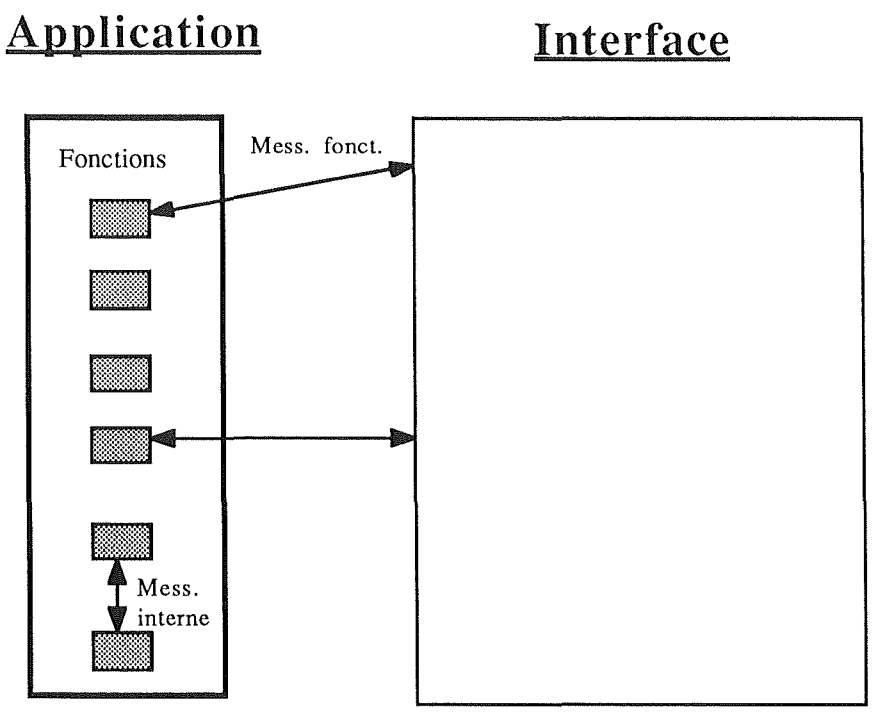


Figure III.2. : Architecture d'une application interactive : version 2

Nous connaissons à présent le moyen de communication entre les fonctionnalités et l'interface d'une application interactive. Nous allons à présent détailler le composant interface.

III.3.Architecture et modélisation de l'interface

L'interface sera composée de trois partie : la Tâche, la Présentation et la Correspondance [F. Bodart & G.Chandelon, 1988].

La Tâche sera composée de messages interactifs sur lesquels l'utilisateur effectuera des opérations sémantiques. Des contraintes d'enchaînement peuvent être posées sur ces opérations.

La Présentation sera la visualisation de la tâche. Les objets interactifs qui la constitueront seront la concrétisation visuelle des messages interactifs; l'utilisateur pourra effectuer sur ces objets des actions physiques qui seront la traduction des opérations sur les messages interactifs. Ces actions devront respecter les mêmes contraintes que celles qui ont été spécifiées pour les opérations.

La correspondance consiste en la traduction des messages fonctionnels en messages interactifs et vice-versa.

Ces trois parties font l'objet des sections suivantes.

III.3.1. Modélisation de la tâche

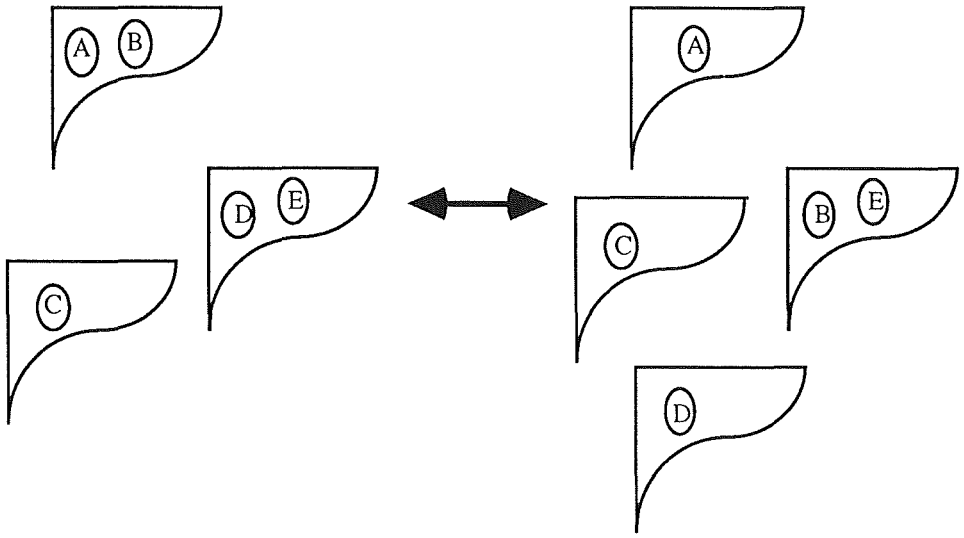
[F. Bodart & G. Warnant, 1988]

Une tâche est un ensemble de messages interactifs sur lesquels l'utilisateur peut effectuer des opérations. Des contraintes d'enchaînement seront spécifiées sur ces opérations.

Nous allons tout d'abord définir un message interactif fonctionnel :

III.3.1.1. Les messages interactifs fonctionnels

Des messages fonctionnels externes ont été définis lors des spécifications fonctionnelles de l'application interactive. Ils ne tiennent cependant pas compte des besoins et des spécificités de l'utilisateur; le regroupement des informations qui les constituent peut être totalement inadapté et doit en conséquence être modifié par l'interface. Les nouveaux messages qui seront créés s'appelent les **messages interactifs fonctionnels**. Chaque information contenue dans un message fonctionnel devra se retrouver une et une seule fois dans un message interactif fonctionnel et vice-versa. Ceci est illustré à la figure III.3.



.Figure III.3. Transposition des messages fonctionnels
en messages interactifs fonctionnels

Un message interactif fonctionnel représente une unité de saisie ou d'affichage pour l'utilisateur. Il considère les informations qui constitue le message comme faisant partie d'un tout; pour lui, le regroupement de ces informations a un sens. Par exemple, un message interactif fonctionnel composé d'une rue, d'un numéro, d'un code postal et d'une localité peut constitué une unité si la réunion de ces informations représente pour l'utilisateur l'adresse d'un client où devront être acheminées les marchandises.

Un message interactif fonctionnel sera de saisie si les informations qu'il contient doivent être fournies par l'utilisateur. Le message est d'affichage si les informations qui le constituent proviennent de l'application proprement dite.

Nous insistons sur le fait qu'il n'existe pas une seule manière de regrouper ou décomposer les informations qui seront manipulées par l'utilisateur; le critère de validation d'un message interactif fonctionnel est qu'il constitue une unité pour celui-ci. Le regroupement ou la décomposition des informations contenues dans les messages fonctionnels devra se faire en fonction des caractéristiques psychologiques de l'utilisateur ou de la classe d'utilisateurs pour qui il sera réalisé[1].

III.3.1.2. Les messages purement interactifs

Les informations contenues dans les messages interactifs fonctionnels permettent l'accomplissement de la tâche de l'utilisateur. Cependant, elles ne lui offrent aucune aide pour y parvenir; elles contiennent le strict nécessaire pour exécuter les fonctionnalités.

Nous avons vu au chapitre II qu'une interface de qualité devait offrir à l'utilisateur des moyens pour se situer par rapport à sa tâche, pour l'aider dans la réalisation de celle-ci en lui fournissant toutes les informations dont il pourrait avoir besoin etc. Ce genre d'informations n'a aucune influence sur les fonctionnalités mais elle favorise son bon déroulement. Elles seront regroupées dans des messages purement interactifs. Ces messages seront communiqués à l'utilisateur sans que les fonctionnalités n'en aient connaissance, puisqu'elles ne les concernent pas.

Le critère de validation d'un message purement interactif est le même que pour un message interactif fonctionnel : il doit constituer une unité de saisie ou d'affichage pour l'utilisateur. La seule différence qui existe entre ces deux types de message interactif tient à l'existence ou l'absence d'une équivalence sémantique (un ou plusieurs messages fonctionnels).

Un message purement interactif sera de saisie si les informations qu'il contient doivent être fournies par l'utilisateur. Un message sera d'affichage si les informations qui le constituent proviennent de l'interface.

III.3.1.3. Les opérations portant sur les messages interactifs

Un message interactif est le moyen de communication entre l'interface et l'utilisateur; ce dernier aura la possibilité de les manipuler pour interagir avec celle-ci. Il le fera à l'aide d'opérations qui porterons soit sur le contenu d'un message (les informations qui le constituent), soit sur sa manipulation parmi un ensemble de messages (sa sélection, son rangement...). Nous détaillerons ces opérations à la section IV.1.4.

III.3.1.4. Les contraintes d'enchaînement

Les opérations qui porteront sur les messages interactifs de l'interface devront en toute généralité respecter un ordre défini. Cet ordre sera spécifié à l'aide de contraintes d'enchaînement. Par exemple, un message interactif fonctionnel de saisie ne pourra être clôturé (la clôture est l'opération par laquelle l'utilisateur signale à l'interface qu'il a fourni des valeurs aux éléments constitutifs du message) que si l'utilisateur a préalablement introduit ces données.

---

[1] Cfr section I.3.4.

Ces contraintes seront spécifiées par le concepteur qui devra prendre soin de les justifier pour qu'elles soient non préjudiciables à l'utilisateur. En effet, une contrainte d'enchaînement restreint la liberté de l'utilisateur en lui interdisant certaintes opérations. Cette limitation doit dès lors être justifiée. Nous définirons plus précisément les contraintes d'enchaînement à la section IV.1.5.

III.3.1.5. Résumé

L'utilisateur effectuera la tâche qu'il s'est fixée et pour laquelle l'application interactive a été prévue en effectuant des opérations sur les messages interactifs qui lui seront présentés (voir figure III.4.). Ces opérations devront respecter des contraintes d'enchaînement.

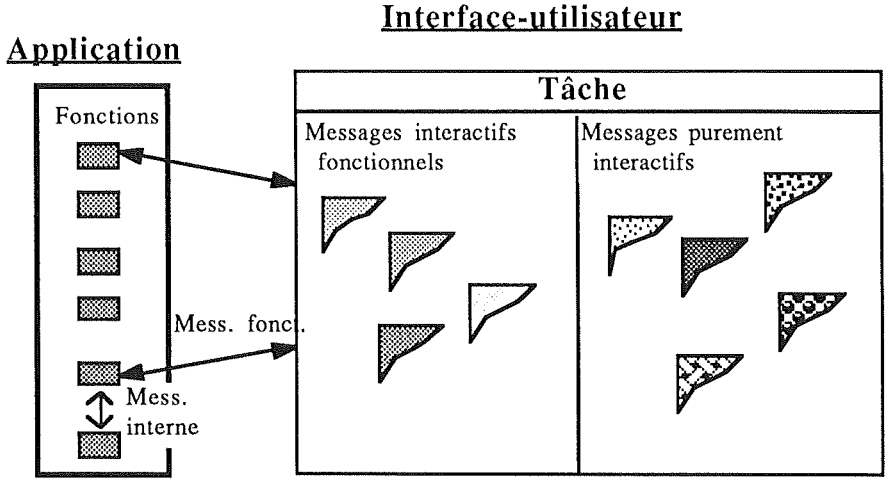


Figure III.4. : Architecture d'une application interactive : version 3

Nous allons à présent définir la visualisation de la Tâche.

III.3.2. La Présentation

Les messages interactifs représentent les informations que l'utilisateur manipulera pour réaliser sa tâche. Ils ne posent aucune hypothèse sur la présentation qui en sera faite. Les objets interactifs sont la visualisation de ces messages à l'écran (ou tout autre appareil d'entrée-sortie) :

III.3.2.1. Les objets interactifs

Un objet interactif présentera à l'utilisateur un ou plusieurs mesages interactifs. Un objet interactif sera par exemple une fenêtre, une icône, un menu, un champ de saisie, un dessin...

Il est essentiel d'apporter la plus grande attention au choix des objets qui apparaîtront à l'écran. En effet, ils constituent les seuls éléments que l'utilisateur verra de l'application interactive. Son acceptation de l'interface se fera tout d'abord en fonction de ce qui lui est présenté. Une application interactive dans laquelle l'utilisateur enregistre la commande d'un client à partir d'un bon de commande sera sans doute appréciée si les objets interactifs affichés reproduisent exactement le bon de commande qu'il doit enregistrer.

Le concepteur de l'interface devra posséder des compétences ergonomiques et psychologiques ou se faire aider par un spécialiste en la matière pour accomplir correctement cette tâche.

III.3.2.2. Les actions physiques

Un message interactif peut être manipulé par des opérations sémantiques. Les actions que pourra effectuer l'utilisateur sur les objets interactifs sont l'équivalent physique de ces opérations. Par exemple, la sélection d'un message se fera en cliquant sur l'objet interactif visualisant le message. Chaque opération aura donc une équivalence physique, représentée par une action.

III.3.2.3. Les contraintes d'enchaînement

Des contraintes d'enchaînement ont été spécifiées sur les opérations portant sur les messages interactifs. Ces contraintes devront bien sûr être répercutées sur les actions physiques qu'effectuera l'utilisateur puisqu'une action est l'équivalent physique d'une opération.

III.3.2.4. Résumé

La Présentation est la concrétisation visuelle de la Tâche. Elle est composée d'objets interactifs présentant à l'utilisateur les messages interactifs. Les objets interactifs peuvent être manipulés par des actions qui sont l'équivalent physique des opérations sur les messages interactifs et doivent de la sorte respecter les mêmes contraintes d'enchaînement. La figure III.5. en est l'illustration.

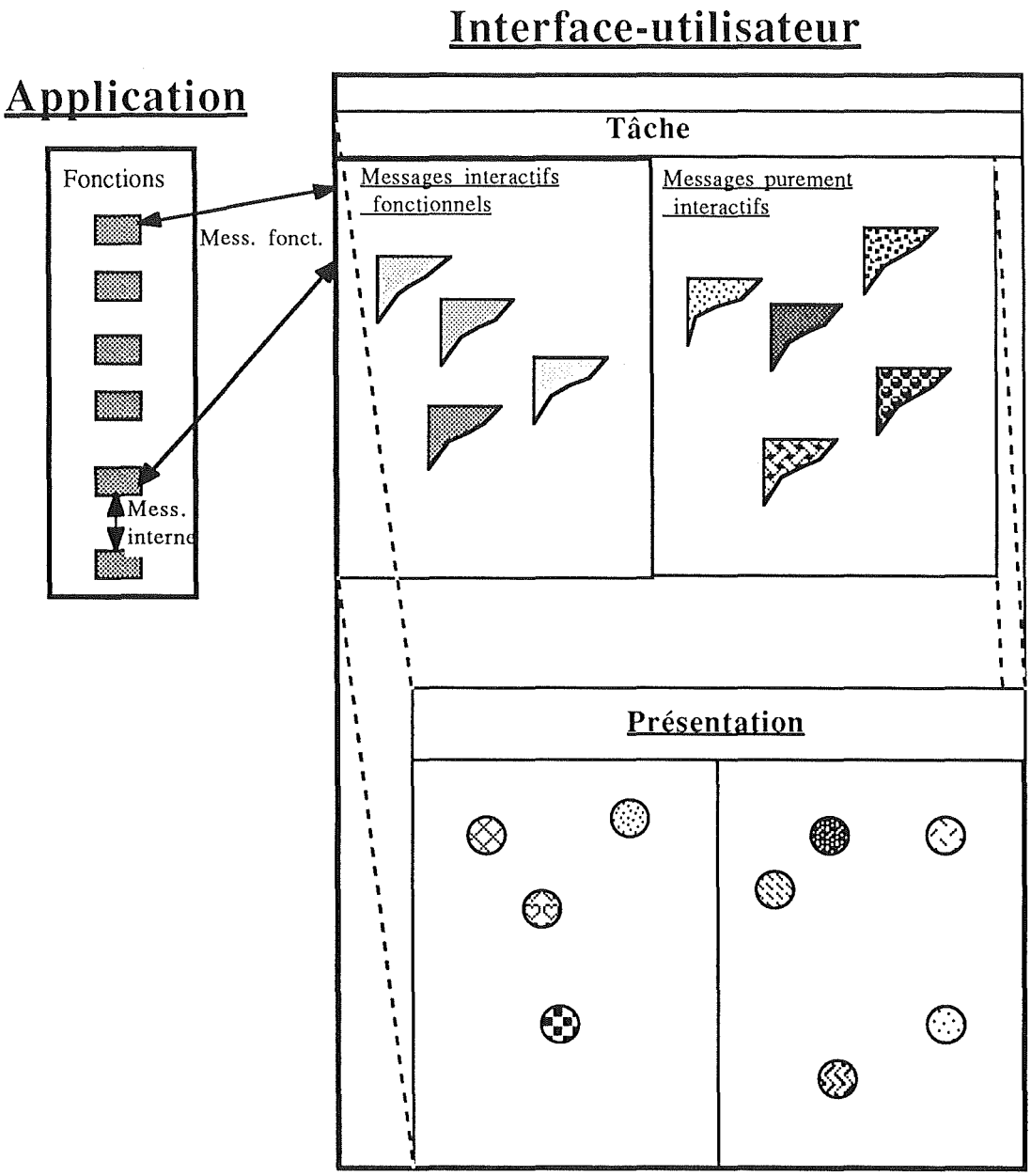


Figure III.5. Architecture d'une application interactive : version 4

III.3.3. La correspondance

Nous avons montré précédemment que les fonctionnalités de l'application interactive communiquaient avec l'interface par l'envoi de messages fonctionnels. D'un autre côté, l'interface fournit à l'utilisateur des messages interactifs fonctionnels pour réaliser les fonctions de l'application. Bien que les informations de ces deux types de messages soient les mêmes[1], leur regroupement est différent. Il est alors nécessaire de disposer d'un mécanisme permettant

[1] Cfr section III.3.1.1.

de traduire les messages fonctionnels en messages interactifs fonctionnels et vice-versa. C'est le but de la correspondance.

La Correspondance constitue le module de traduction entre l'interface et les fonctionnalités; tout message fonctionnel provenant de l'application proprement dite sera traduit en terme de message interactif fonctionnel et le même principe s'applique à un message provenant de l'interface.

Nous avons choisi de la représenter comme un élément de l'interface. En effet, nous avons vu l'importance de l'indépendance des fonctionnalités par rapport à l'interface; celles-ci ne doivent pas connaître la présentation qui sera faite des éléments qu'elles manipulent. Il n'était donc pas question de placer ce module dans l'application proprement dite. Il eut été possible d'ajouter un composant intermédiaire entre l'interface et les fonctionnalités. Cependant, nous considérons que la traduction des messages entre ces deux composantes fait partie intégrante du rôle de l'interface. D'un côté, elle traduit les actions de l'utilisateur et en fait part aux fonctions; de l'autre, elle traduit les informations de l'application proprement dite et en informe l'utilisateur. La correspondance est un élément de cette double traduction.

Ceci est représenté à la figure III.6.

III.3.4. Modélisation du dialogue

Les divers éléments de l'interface que nous venons de spécifier peuvent être représentés sous forme d'un schéma entités/associations [F. Bodart & G. Warnant, 1988] :

L'utilisateur exécute une ou plusieurs tâches visualisées par une ou plusieurs présentations. Une tâche est composée de messages interactifs sur lesquels l'utilisateur effectue des opérations qui doivent respecter des contraintes d'enchaînement. Les objets interactifs sont la représentation des messages interactifs à l'écran. L'utilisateur réalisera sur ces objets des actions physiques qui devront respecter les mêmes contraintes que celles qui ont été imposées aux opérations sur les messages interactifs.

La modélisation est présentée à la figure III.7.

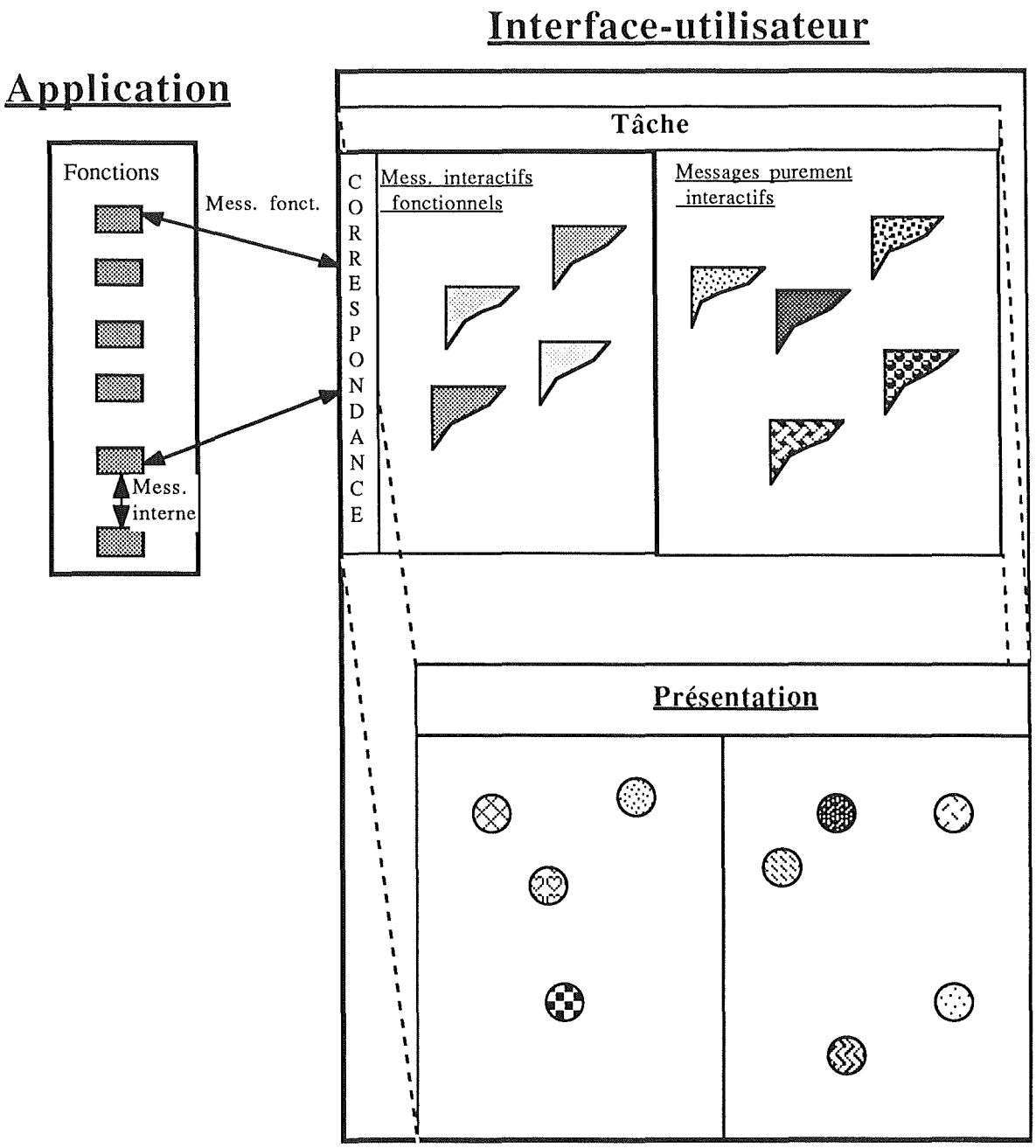


Figure III.6. : Architecture d'une application interactive : version définitive

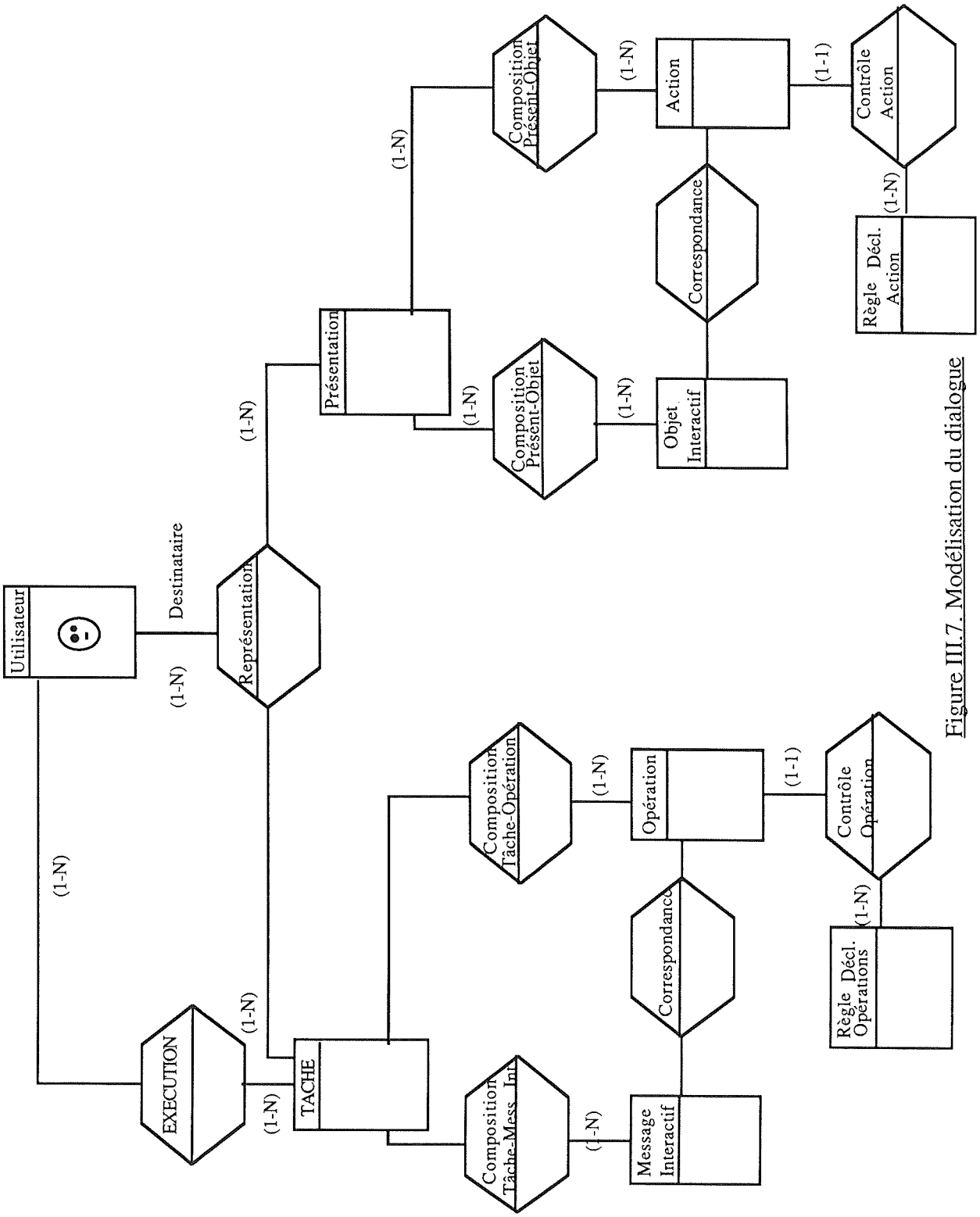


Figure III.7. Modélisation du dialogue

III.4. Présentation de la méthodologie générale

Le principe de séparation modulaire entre l'interface-utilisateur et les fonctionnalités de l'application interactive nous permet de faire une séparation nette entre leur spécification.

La première étape de notre méthodologie générale sera l'analyse des besoins et l'étude d'opportunité. Ensuite viendra l'analyse fonctionnelle de l'application interactive à réaliser. Nous rappelons que la méthodologie de F. Bodart et Y. Pigneur sera utilisée pour cette étape. Elle peut donner lieu à un prototypage ou à une simulation en vue de tester la validité des traitements qui seront effectués, l'impact sur les ressources disponibles...

Une fois les spécifications fonctionnelles validées, l'étape suivante consiste en la spécification de l'interface-utilisateur. Certaines informations provenant de la phase précédente seront nécessaires. Elles font partie de la statique et de la dynamique des traitements. La statique permettra de connaître les messages fonctionnels qui ont été définis et l'objectif des différents traitements; la dynamique spécifiera "*les conditions de déclenchement et d'enchaînement des traitements*" (Bodart & Pigneur, 1988). Nous avons montré dans les sections précédentes qu'on pouvait modéliser le dialogue qui avait lieu entre l'utilisateur et l'application interactive au moyen de deux concepts : la tâche à effectuer et la présentation qui en sera faite. La spécification de l'interface-utilisateur se décomposera tout naturellement en une spécification de la Tâche et une spécification de la Présentation.

Cette séparation permet au concepteur de ne pas s'attacher directement aux aspects de la présentation de son interface pour ne s'intéresser qu'aux informations nécessaires à la réalisation de sa tâche ainsi qu'à leur regroupement et leur enchaînement.

La spécification de la Tâche comprendra dans un premier temps la spécification des messages interactifs (unités de saisie ou d'affichage pour l'utilisateur) qui seront présentés à l'utilisateur. Nous définirons ensuite les opérations qui peuvent s'effectuer sur ces messages et leurs contraintes d'enchaînement.

Dans le but d'éviter des incohérences entre les spécifications fonctionnelles et la spécification de la Tâche, suivra une phase de validation où l'on vérifiera que les informations ayant un impact sémantique se retrouvent une et une seule fois dans les spécifications fonctionnelles et vice-versa. Ensuite, il nous a semblé utile de permettre au concepteur d'obtenir un schéma intégrant la spécification de la tâche et la spécification des traitements découlant de la phase de spécifications fonctionnelles. Ceci est réalisé par le schéma de la dynamique d'implémentation.

Enfin viendra la spécification de la Présentation. Nous y spécifierons en premier lieu les objets interactifs présentant les messages interactifs. Nous définirons les actions physiques que peut réaliser l'utilisateur et leurs contraintes d'enchaînement.

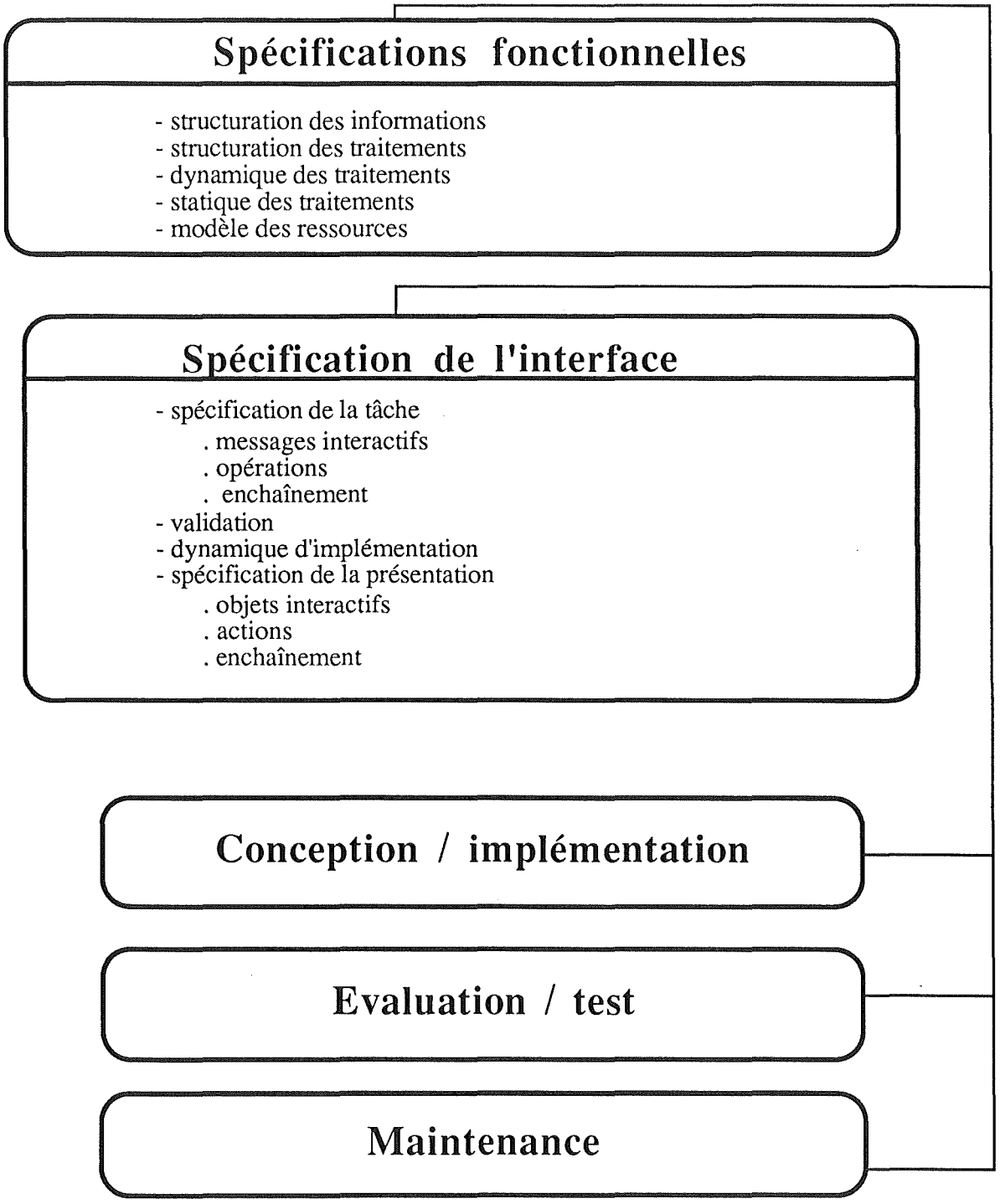


Figure 3.8. : Méthodologie générale de conception d'une application interactive

Ces différentes phases seront exposées plus longuement dans le chapitre suivant. Notre but ici était de justifier notre approche d'un point de vue global.

Ceci nous donne le schéma de méthodologie de conception d'une application interactive présenté à la figure III.8. Les étapes qui suivent les spécifications sont là pour mémoire. Rappelons que ce travail portera uniquement sur la spécification de l'interface-utilisateur.

III.5. Conclusion

La méthodologie générale de conception d'une application interactive que nous venons de développer, propose une extension de la méthodologie développée par F. Bodart et Y. Pigneur. Elle s'appuie sur le principe de séparation modulaire entre les fonctionnalités et l'interface. La phase de spécification des fonctionnalités sera dès lors suivie de la spécification de l'interface.

Cette dernière se décompose en l'analyse de la Tâche et de la Présentation. La Tâche spécifie le contenu des informations communiquées à l'utilisateur et les opérations par lesquelles celui-ci les manipulera. La Présentation sera la visualisation de la Tâche à l'écran.

La spécification de l'interface se basera sur les résultats de l'analyse fonctionnelle et d'une étude approfondie du ou des utilisateurs futurs.

Le chapitre suivant présente en détails les éléments qui permettront au concepteur de spécifier le dialogue de l'application intéractive.

Chapitre IV
Méthodologie de spécification de l'interface-utilisateur

IV.0. Introduction

Tandis que les deux premiers chapitres situaient le contexte de ce travail, le chapitre III posait les fondements de notre méthodologie.

Ce chapitre a pour but de présenter plus en détail la méthodologie que nous proposons pour spécifier une interface homme-machine. Des choix concernant les informations à spécifier et l'ordre dans lequel elles le seront ont dû être faits. Ils seront justifiés dans l'exposé et testés dans le chapitres VI, traitant de l'évaluation de la méthodologie.

Nous suivrons, étape par étape, ce qui a été proposé à la section III.4. du chapitre précédent. Nous commencerons par l'analyse de la Tâche. La distinction entre message purement interactif et message interactif fonctionnel conduira à la séparation de leur spécification. Nous analyserons ensuite les éléments de spécification d'un message interactif fonctionnel avant d'étudier les opérations que l'on peut lui associer. Nous analyserons alors l'expression des contraintes d'enchaînement de ces opérations en présentant le schéma de la conversation. Puis, nous spécifierons les messages purement interactifs.

L'étape suivante sera la validation de l'analyse de la Tâche par rapport à la dynamique des traitements. Nous intégrerons alors le schéma de la conversation à la dynamique des traitements pour produire le schéma de la dynamique d'implémentation.

La dernière partie de notre méthodologie de spécification de l'interface-utilisateur sera la spécification de la Présentation. Nous analyserons les informations nécessaires à la spécification des objets interactifs, les actions physiques que l'utilisateur peut effectuer et enfin les contraintes d'enchaînement de ces dernières.

Nous débutons par l'analyse de la tâche.

IV.1. Analyse de la Tâche

"La Tâche est l'expression fonctionnelle des actions que l'utilisateur devra réaliser pour mener à bien l'exécution de l'application interactive. Une tâche est composée de messages interactifs sur lesquels l'utilisateur peut effectuer des opérations sémantiques à propos desquelles on impose une série de contraintes d'enchaînement" [F. Bodart & G. Warnant, 1988].

La section suivante analyse les conséquences de la différence entre un message purement interactif et un message interactif fonctionnel.

IV.1.1. Distinction entre message interactif fonctionnel et message purement interactif

Un message interactif fonctionnel contient des informations manipulées par les fonctionnalités tandis que le contenu d'un message purement interactif n'intéresse que l'utilisateur; l'application proprement dite n'en a pas connaissance[1]. Nous pouvons décomposer les messages purement interactifs en 4 classes :

- les **messages de contrôle** permettent à l'utilisateur de se déplacer dans le schéma de la conversation. Ce schéma représente les contraintes d'enchaînement qui seront spécifiées sur les opérations de l'utilisateur. Par exemple, l'utilisateur aura la liberté de saisir un message interactif fonctionnel correspondant au nom d'un client ou un autre correspondant à son numéro. Il informera l'interface de son choix en sélectionnant dans le message de contrôle l'information qui y correspondra. Le schéma de la conversation fait l'objet de la section IV.1.3.

- **les messages d'aide** représentent les informations que l'interface homme-machine dispense à l'utilisateur pour l'aider dans la réalisation de sa tâche. Il s'agit d'informations concernant les fonctionnalités de l'application interactive, mais aussi du contexte dans lequel se trouve l'utilisateur, de l'état du système à un moment donné etc. Ces messages devraient constituer une véritable aide généralisée, un tableau de bord susceptible de répondre à toutes les questions qu'une personne en interaction avec l'application peut se poser.

- les **messages d'erreur syntaxique** sont définis ici dans une acception plus large que celle qui est généralement utilisée. Une erreur syntaxique est toute erreur n'ayant pas été relevée dans l'analyse fonctionnelle mais qui est susceptible de se produire selon l'implémentation qui sera faite des fonctionnalités. Par exemple, dans un environnement de manipulation directe, l'utilisateur n'entrant pas toutes les données nécessaires à la réalisation d'un traitement verrait apparaître un message d'erreur syntaxique l'informant de l'incomplétude. Ce type d'erreur porte sur les contraintes d'enchaînement des opérations sur les messages interactifs fonctionnels, c'est-à-dire sur le schéma de la conversation. Ce dernier permettra de définir le scénario de traitement des messages que l'utilisateur devra respecter[2]. Il est important de noter que les choix qui seront faits quant aux objets interactifs présentant les messages, ainsi que le degré de liberté qui sera laissé à l'utilisateur, ont une grande influence sur les messages d'erreur syntaxique qui devront exister. Nous reportons le lecteur aux sections suivantes pour plus de détails.

- les **message de données propres au dialogue** permettent à l'utilisateur de choisir, pour une même application interactive, parmi plusieurs interfaces-utilisateur possibles. Ce type de message dépasse le cadre de spécification d'une interface bien particulière. Nous ne reviendrons donc plus sur ce type de message.

A partir de la définition des différents types de message purement interactifs qui précède, nous avons décidé d'attendre la fin de la spécification du schéma de la conversation pour spécifier les messages purement interactifs. En effet, les messages de contrôle liés à la conversation dépendent bien évidemment de celle-ci. Nous avons vu également que certains

---

[1] Cfr section III.3.1.

[2] Voir section IV.1.3.

messages d'erreur syntaxique pouvait en dépendre. Ceci nous permet en outre de dissocier parmi les messages interactifs ceux qui ont un impact sur les fonctionnalités et en dépendent directement (les messages fonctionnels interactifs) de ceux qui concernent uniquement l'interface-utilisateur (les messages purement interactifs).

En résumé, l'analyse de la tâche comprendra les étapes suivantes :

1° La spécification des messages interactifs fonctionnels;
2° L'expression des contraintes : le schéma de la conversation;
3° La validation de l'analyse de la tâche par rapport à la dynamique des traitements;
4° La dynamique d'implémentation;
5° La spécification des messages purement interactifs.

Nous pouvons à présent passer à la première étape de l'analyse de la Tâche : la spécification des messages interactifs fonctionnels.

IV.1.2. Spécification des messages interactifs fonctionnels

Un message interactif fonctionnel contient des informations qui ont été relevées lors de l'analyse fonctionnelle de l'application interactive. Les informations qui le constitueront feront partie des messages fonctionnels développés lors de la phase d'analyse fonctionnelle. Nous avons décidé de le spécifier à l'aide d'une liste d'éléments dont voici la définition :

- **NOM** : Le nom donné au message interactif fonctionnel.

- **DEFINITION** : La définition du message fonctionnel interactif, en langage naturel.

- **TYPE** : Un message peut être d'affichage ou de saisie. S'il est d'affichage, il sera affiché (ou plus précisément le ou les objets interactifs qui le présenteront) par l'interface-utilisateur et proviendra de l'application proprement dite. S'il est de saisie, il sera rempli par l'utilisateur qui le communiquera, via l'interface, aux fonctionnalités de l'application interactive après la phase de traduction réalisée par la Correspondance[1]. Un message ne pourra pas être à la fois d'affichage et de saisie : dans le cas où un même regroupement d'informations peut être communiqué par l'utilisateur ou l'interface (par exemple, l'interface communique l'adresse du client passant la commande et l'utilisateur a la possibilité de la modifier en cas de déménagement etc.), nous distinguerons le message d'affichage (communiqué par l'interface) du message de saisie (la modification qu'a effectuée l'utilisateur). Ces deux messages contiennent les mêmes éléments et seront présentés à l'écran par un même objet interactif[2].

- **JUSTIFICATION** : Nous avons déjà insisté sur l'importance qu'il fallait accorder à l'unité que doit représenter un message interactif pour l'utilisateur. Le spécificateur argumentera ici les raisons qui l'ont poussé à considérer ce message comme une unité de saisie ou d'affichage pour l'utilisateur;

---

[1] Cfr section III.3.3.

[2] Voir section IV.4.

- **STRUCTURE DE DONNEES** : La définition des éléments constitutifs du message (les attributs), de leur type ainsi que des contraintes d'intégrité pouvant y être attachées. Le type et les contraintes d'intégrité devront correspondre à ce qui a été défini pour ces mêmes attributs lors des spécifications fonctionnelles.

- **CONTRAINTES SUR LA PRESENTATION** : Ces contraintes définissent les aspects ergonomiques ou psychologiques dont le spécifieur voudrait qu'il soit tenu compte. En effet, des contraintes sur la présentation peuvent avoir été formulées lors d'une discussion avec l'utilisateur futur, ou encore l'analyseur de la Tâche peut être un spécialiste des aspects cognitifs ou ergonomiques qui veut donner des directives au concepteur de la Présentation etc. En bref, il nous a semblé utile de pouvoir, dès la phase d'analyse de la Tâche, spécifier les contraintes dont on désirerait qu'il soit tenu compte dans la présentation des messages interactifs. De la sorte, les informations nécessaires au bon choix des objets interactifs seront présentes.

- **OPERATIONS DE MANIPULATION** : La définition des opérations que l'utilisateur pourra réaliser sur le message interactif. N'oublions pas que ce message sera en fait visualisé par un objet interactif auquel on associera des actions qui devront être la traduction des opérations que nous sommes en train de définir. Ces opérations font l'objet de la section suivante.

IV.1.3. Opérations sur les messages interactifs fonctionnels

Nous pouvons distinguer deux types d'opération sur les messages interactifs fonctionnels : les opérations aidant le spécifieur dans la création de ses messages et les opérations aidant l'utilisateur à manipuler ses messages (les créer, les remplir...).

IV.1.4.1. Opérations primitives de définition d'un type de message

Une opération primitive de définition d'un type de message permet au spécifeur de définir un message, éventuellement en fonction de messages préalablement spécifiés. On peut imaginer de la sorte une librairie de messages fréquents parmi lesquels le spécifieur choisira ceux qui lui sont appropriés.

Ces opérations sont telles que le nommage, le renommage, l'union, l'intersection, l'éclatement... de messages. Elles sont analogues à celles de l'algèbre relationnel.

Ce type d'opération ne nous intéresse que dans une moindre mesure car nous voulons tout d'abord fournir au concepteur toutes les informations nécessaire à la spécification de son interface et non pas déjà des mécanismes pour faciliter sa réalisation. Nous reportons le lecteur intéressé à Collet (1987).

IV.1.4.2. Opérations primitives de manipulation d'un type de message

Une opération primitive de manipulation d'un type de message permet à l'utilisateur d'opérer sur un message ou sur son contenu. On peut ainsi distinguer les primitives liées au contenu d'une occurence d'un message, c'est-à-dire permettant à l'utilisateur de manipuler les éléments constitutifs du message, et les primitives liées à sa manipulation parmi l'ensemble des messages accessibles à l'utilisateur. Nous détaillons à présent ces deux types d'opération primitive.

a) Primitives liées au contenu d' une occurence d'un message

Ces opérations portent sur les attributs - ou les champs selon la terminologie employée - des messages. Les opérations que l'on peut associer sont :

- la **sélection** d'un attribut d'un message : l'utilisateur sélectionne un attribut parmi les attributs du ou des messages;

- l'**affectation** d'une valeur à un attribut d'un message : l'utilisateur introduit une valeur pour l'attribut qu'il a préalablement sélectionné;

- la **suppression** de la valeur d'un attribut d'un message : l'utilisateur supprime la valeur que contenait l'attribut qu'il a sélectionné;

- la **correction** de la valeur d'un attribut d'un message : l'utilisateur modifie la valeur que contenait l'attribut qu'il a sélectionné;

- l'**activation d'une contrainte** portant sur un ou plusieurs attributs du message : l'utilisateur demande la vérification d'une contrainte que doivent remplir un ou plusieurs attributs du message.

L'opération de **sélection** est indispensable lorsque l'utilisateur peut choisir d'opérer sur plusieurs attributs. Cela sera certainement le cas dans un environnement de manipulation directe mais également avec des styles de dialogue plus traditionnels comme le remplissage de formulaire (en utilisant une opération de sélection pour voyager d'un attribut à un autre).

Pour ce qui est des **contraintes** portant sur un ou plusieurs champs d'un message interactif fonctionnel, il est important de définir quelles sont les contraintes qui doivent être prises en compte par l'interface-utilisateur et celles qui doivent être laissées à l'application proprement dite. En d'autres termes, où doit-on poser la frontière entre ce qui doit être vérifié par l'application proprement dite et ce qui doit l'être par l'interface-utilisateur?

Dans le but de réaliser un maximum de validation par l'interface-utilisateur, nous prendrons comme critère celui de l'accès à la base de données. Toute contrainte ne nécessitant pas un accès à la base de données sera validée par l'interface-utilisateur. Nous avions déjà relevé lors de la définition des messages purement interactifs de type erreur syntaxique que nous considérions ce type d'erreur dans un sens plus étendu que celui qui est généralement utilisé. De la sorte, toute erreur enregistrée par l'interface-utilisateur débouchera sur une erreur syntaxique, tandis que tout erreur détectée par les fonctionnalités de l'application proprement dite déclenchera une erreur sémantique (une erreur sémantique provoquera l'envoi d'un message fonctionnel).

Nous avons préalablement parlé de deux types de message interactif : les messages de saisie et les messages d'affichage. Il est clair qu'un message d'affichage ne fait que présenter des informations à l'utilisateur. Une fois que l'application l'a envoyé à l'interface-utilisateur, elle n'en attend aucune contrepartie. Par contre, un message de saisie est rempli par l'utilisateur et ensuite envoyé à l'application qui en a besoin pour ses traitements.

Ainsi, les primitives que nous venons de définir concerneront uniquement les messages de saisie puisqu'elles n'ont aucun sens pour un message d'affichage qui présente simplement de l'information non manipulable par l'utilisateur.

Parmi les messages de saisie, on peut encore faire une distinction entre deux types d'attribut : un attribut est soit facultatif, soit obligatoire. Il est facultatif s'il n'est pas indispensable qu'il ait une valeur lors de la saisie du message. On pourra par exemple ne pas exiger l'introduction de la profession lors de la saisie des informations concernant un client. L'utilisateur aura la possibilité de ne pas affecter de valeur à cet attribut sans que cela n'entraîne un message d'erreur syntaxique. Un attribut est obligatoire s'il est indispensable à l'application et doit nécessairement être saisi. De cette distinction, on peut conclure que :

- Un attribut **obligatoire** d'un message de **saisie** peut être manipulé par les primitives **d'affectation**, **correction**, **sélection** et **activation**;

- Un attribut **facultatif** d'un message de **saisie** peut être manipulé par les primitives **d'affectation**, **suppression**, **correction**, **sélection** et **activation**;

- Un message **d'affichage** ne peut pas être manipulé par ces primitives.

b) Primitives liées à la manipulation d'un message parmi un ensemble de messages

Ces opérations concernent la manipulation par l'utilisateur d'un message parmi un ensemble de messages possibles. Ainsi, on définit :

- La **sélection** d'une occurence d'un message;
- La **création** d'une occurence d'un message;
- La **suppression** d'une occurence d'un message;
- Le **rangement** d'une occurence d'un message (la disparition du message de l'écran);
- La **clôture** d'une occurence d'un message (confirmation de la terminaison de la saisie).

Un message sera **créé** et **supprimé** si sa durée de vie est dépendante de l'application interactive. Ce message fait alors partie du code de l'application et n'est pas accessible en dehors de celle-ci. Il sera créé lorsque s'exécutera les instructions du programme prévues pour l'afficher et sera détruit lorsqu'il disparaîtra de l'écran. Au contraire, un message pourra être **sélectionné** et **rangé** s'il fait partie d'un ou plusieurs fichiers de ressources gérés par l'application interactive. Un fichier de ressources contiendra l'enregistrement d'un ensemble des informations qui seront communiquées à l'utilisateur. Grâce à ce type de fichiers, on pourra :

- **Créer plusieurs formats pour une même information.** Il sera par exemple possible de fournir les mêmes informations dans plusieurs langues; l'utilisateur indiquera la langue qu'il désire et l'interface présentera les données qui y correspondent.

- **Modifier les informations en dehors de l'application interactive.** L'utilisateur pourra les modifier selon ses désirs (l'interface devient adaptable).

- **Réutiliser des messages pour différentes interfaces ou différentes applications interactives.**

En résumé, nous faisons la distinction entre les messages créés dynamiquement lors du déroulement de l'application et les messages ayant une durée de vie indépendante de l'application interactive.

La **clôture** concerne uniquement les messages de saisie. Par cette opération, l'utilisateur informe l'interface-utilisateur qu'il a terminé de saisir le message sur lequel elle porte.

IV.1.4.3. Résumé

L'utilisateur peut effectuer des opérations sur les messages interactifs que lui communique l'interface. Ces opérations portent sur le contenu d'un message ou sur sa manipulation parmi un ensemble. Elles feront partie de la spécification des messages interactifs. Des contraintes peuvent être posées sur leur enchaînement. Nous les analysons à la section suivante.

IV.1.4. Expression des contraintes

IV.1.4.1. Introduction

Nous avons jusqu'à présent spécifié les messages fonctionnels interactifs. Nous avons vu qu'on pouvait leur associer des primitives liées à leur contenu ou à leur manipulation parmi un ensemble de messages. Nous allons maintenant nous pencher sur le problème de l'expression des contraintes d'enchaînement des opérations qui leur sont associées.

Nous avons, lors des sections précédentes, précisé les opérations que pouvait effectuer un utilisateur sur les messages interactifs fonctionnels en fonction de leurs caractéristiques (de saisie ou d'affichage, permanent ou non, ...). Le spécifieur définira ces opérations dans la spécification des messages. Il connaîtra donc les opérations qui seront autorisées sur chaque message. Il lui reste à spécifier l'ordre qu'elles decront respecter. Nous allons voir que celui-ci sera en partie implicite :

a) Primitives liées au contenu

Nous pouvons énoncer les règles suivantes :

- La **sélection** d'un attribut doit précéder son **affectation**, sa **suppression** ou sa **correction**. Un attribut doit en effet être désigné avant de pouvoir effectuer toute opération sur lui[1];
- l'**affectation** d'une valeur à un attribut doit précéder la **suppression** et la **correction** de cette valeur. Un attribut doit posséder une valeur pour qu'elle puisse être supprimée ou corrigée;
- l'**affectation** d'une valeur à un attribut doit préceder l'**activation des contraintes** pour tous les attributs sur lesquels elle porte. Les contraintes sur un ou plusieurs attributs ne peuvent être vérifiées si certains de ces attributs ne possèdent pas de valeur.

b) Primitives liés à la manipulation parmi un ensemble de messages

Nous pouvons énoncer les règles suivantes :

- Un message ne pourra être **supprimé** que s'il a été précédemment **créé**.
- Un message ne pourra être **rangé** que s'il a été précédemment **sélectionné**.
- Un message ne pourra être **clôturé** que s'il a été précedemment **créé** ou **sélectionné** et s'il est de **saisie**.

L'ordre des opérations que pourra effectuer l'utilisateur sur un message interactif fonctionnel devra respecter ces différentes règles. Cet ordre peut dès lors être considéré comme implicite.Par contre, l'enchaînement des opérations qu'il pourra réaliser sur l'ensemble des messages doit être spécifié. Cet enchaînement revient à définir l'ordre dans lequel l'utilisateur pourra agir sur l'ensemble des messages. Il devra par exemple attendre d'avoir traité un message M1 avant de pouvoir opérer sur un message M2. L'expression de cet enchaînement sera représenté par le schéma de la conversation. On y spécifiera l'ordre de traitement des messages que l'utilisateur devra respecter.

Il est important de souligner à nouveau que lors de la spécification des messages interactifs, le spécificateur doit faire des choix de conception. En effet, il est toujours possible de choisir une autre découpe des messages interactifs fonctionnels. Il en va de même pour la spécification du schéma de la conversation. Le spécifieur pourra choisir de laisser à l'utilisateur un maximum de degré de liberté, par exemple si l'utilisateur futur est un expert dans la tâche, ou encore de lui imposer un ordre de marche, si l'utilisateur est un novice ou si un critère d'efficacité l'impose. Un haut degré de liberté laissera à l'utilisateur un large choix parmi les messages qu'il pourra traiter à un moment donné tandis qu'un ordre de marche sévère lui imposera le message sur lequel il devra opérer.

On pourra de cette façon spécifier plusieurs schémas de la conversation, qui correspondront à différentes interfaces pour une même application .

[1] Cette règle n'est valable que dans le cas où cet attribut peut être sélectionné. Si l'interface oblige l'utilisateur à traiter l'attribut, il est clair que cette règle ne pourra être respectée.

Les concepts qui seront utilisés dans la spécification de la conversation sont proches de ceux du schéma de la dynamique des traitements décrit dans [F. Bodart et Y. Pigneur (1988). Nous les présentons ici en termes généraux :

IV.1.4.2. Concepts du schéma de la conversation

Les concepts que nous présentons ici sont une tentative d'extension du formalisme décrivant la dynamique des traitements dans la méthodologie de F. Bodart et Y. Pigneur. Nous n'avons pas la prétention de croire que cette extension est complète et permet de représenter complètement l'interaction qui aura lieu entre l'interface homme-machine et l'utilisateur. Nous la testerons et l'évaluerons au chapitre VI. Notre objectif est de proposer un formalisme proche des diagrammes de transition d'état. Il permettra de spécifier l'enchaînement des opérations qui pourra être réalisée par l'utilisateur lorsqu'il exécutera l'application interactive.

Les états que nous distinguerons correspondront à la fin du traitement d'un message interactif. Un état sera représenté par un point situé au bas du message traité. Le déclenchement des opérations qui devront le suivre sera représenté par une flèche. Par exemple, une flèche dont l'origine est le point du message M1 et la destination le message M2 signifiera que la fin du traitement du message M1 (représenté par le point) déclenche (la flèche) le traitement du message M2.

Nous présentons différentes structures d'enchaînement introduites par G. Warnant [F. Bodart & G. Warnant, 1988) et qui devraient permettre de spécifier complètement la conversation entre l'interface et l'utilisateur. Nous analyserons au chapitre VI la validité de ce formalisme.

a) le parallélisme

La structure parallèle est un type d'enchaînement par lequel le concepteur offre la possibilité à l'utilisateur de traiter une liste de messages sans présumer de l'ordre qui sera choisi par ce dernier pour y parvenir. La fréquence d'utilisation de cette structure déterminera le degré de liberté qui sera offert.

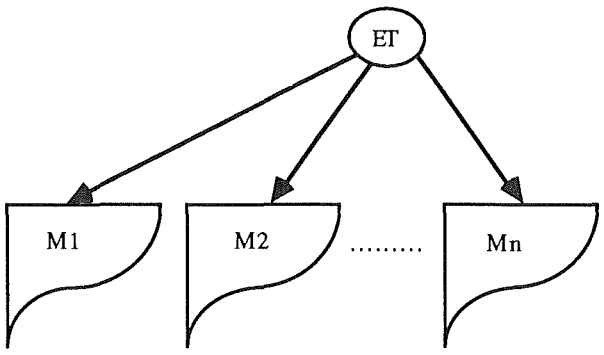


Figure IV.1. : structure parallèle

b) L'enchaînement séquentiel

Un enchaînement de messages est séquentiel lorsque la terminaison de la saisie ou de l'affichage d'un message M1 provoque le déclenchement de la saisie ou de l'affichage d'un message M2. Si M2 est un message d'affichage, il n'apparaîtra à l'écran qu'à la terminaison de M1. S'il est de saisie, l'utilisateur ne pourra le saisir qu'après avoir terminé le traitement du message M1.

La fréquence d'utilisation de cette structure aura l'effet inverse de la structure paralèlle. Elle imposera un ordre de marche à l'utilisateur.

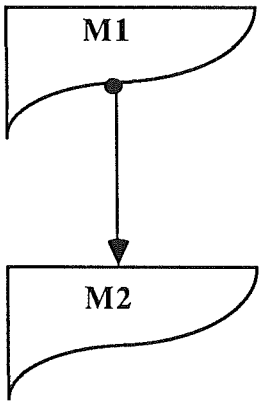


Figure IV.2. : Enchaînement séquentiel

c) La structure convergente

Un enchaînement de messages est convergent si le déclenchement de la saisie ou de l'affichage d'un message Mi est provoqué par la terminaison de la saisie ou de l'affichage de l'un quelconque des messages M1, M2, ..., Mn.

Un enchaînement convergent est la réunion de n enchaînements séquentiels déclenchant un même message Mi.

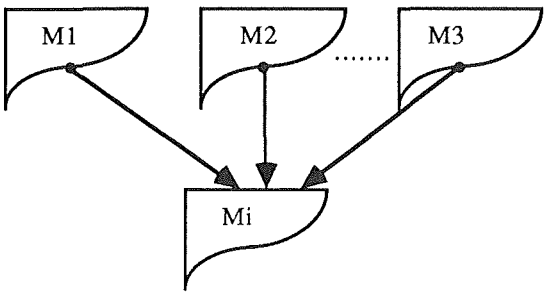


Figure IV.3. : L'enchaînement convergent

d) La synchronisation de messages

Un mécanisme de synchronisation est nécessaire pour permettre au concepteur de spécifier l'enchaînement d'un ou plusieurs messages suite à la terminaison du traitement de deux messages ou plus. C'est *"un mécanisme de coordination d'évènements"* [Bodart & Pigneur, 1983]. Par exemple, la terminaison de la saisie ou de l'affichage d'un message M1 et d'un message M2 provoquera le déclenchement du traitement d'un message M3 (voir figure IV.4.)

On peut distinguer deux types de synchronisation :

- La **conjonction** : les occurences de message qui participent à la synchronisation sont de types différents[1]. Un type de message représente ici le nom qui lui est associé. Par exemple, un message "commande_saisie" constitue un type de message dont les occurences sont les différents messages correspondant aux commandes qui ont été saisies. Ce type de synchronisation est représenté par un trapèze. La contribution de chaque occurence de message qui participe à la synchronisation sera représentée par une flèche au départ du message et à destination du trapèze. Les flèches quittant le trapèze représenteront le déclenchement du traitement des messages qui doit être effectué lorsque la synchronisation a été réalisée.

- L'**accumulation** : les occurences de message qui participent à la synchronisation sont de même type. Il s'agira par exemple de saisir n messages " Ligne_de_commande" avant d'afficher le message "Total_commande". Ce type de synchronisation est représenté par un triangle renversé; un exemple est donné à la figure IV.5.

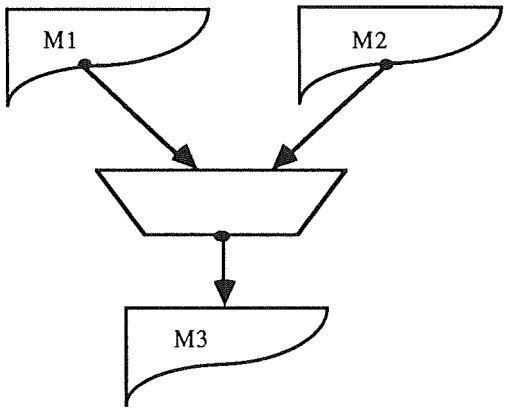


Figure IV.4. : Synchronisation de deux processus

[1] Ou plus précisément, il ne sont pas tous du même type.

e) L'itération de messages

Une structure est itérative si la terminaison de la saisie ou de l'affichage d'un message M1 déclenche la saisie ou l'affichage de plusieurs occurences d'un message M2. La structure itérative peut porter sur un ou plusieurs messages; la terminaison d'un message M1 pourra déclencher la saisie ou l'affichage d'un certain nombre de message M2, M3, ..., Mn.

Pour délimiter les messages sur lesquels porte l'itération, un signe de fin d'itération est représenté qui signifie que la totalité des messages sur lesquels l'itération portait doit avoir été traitée. La fin d'itération est donc un type particulier de synchronisation qui correspond à l'accumulation de messages que nous avions définie au paragraphe précédent. La représentation de ce type de synchronisation est un triangle renversé.

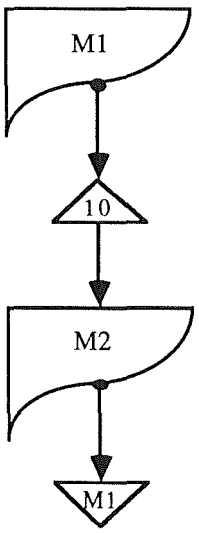


Figure IV.5. : L'itération de message : au message M1 succède 10 occurences du message M2.
Ces 10 occurences du message M2 participent à la synchronisation représentée par le triangle renversé.
Ce type de synchronisation est appelé l' accumulation.

Le nombre d'itération est représenté dans le triangle de début d'itération. Il peut s'agit d'une constante ou d'une condition à vérifier. Cette condition peut être de complexité quelconque : elle peut porter sur une valeur saisie, être calculée pendant l'itération courante... Elle peut également dépendre de l'utilisateur[1]. L'itération que nous décrivons ici est tout-à-fait similaire à une boucle "while" dans un langage de commande.

f) la structure conditionnelle

La structure conditionnelle est un type d'enchaînement par lequel la fin de la saisie ou de l'affichage d'un message M1 provoque le déclenchement de la saisie ou de l'affichage d'un message M2 ou M3, ..., ou Mn. La condition de déclenchement sera représentée par un losange dans lequel on spécifiera la condition qui doit être évaluée. Cette condition peut correspondre au

[1] Par exemple, le nombre de lignes que comporte une commande sera déterminé par l'utilisateur.

choix qui est proposé à l'utilisateur d'effectuer l'un ou l'autre traitement[1]. Des flèches sur lesquelles on définira le résultat de la condition partiront de celui-ci à destination des messages à traiter.

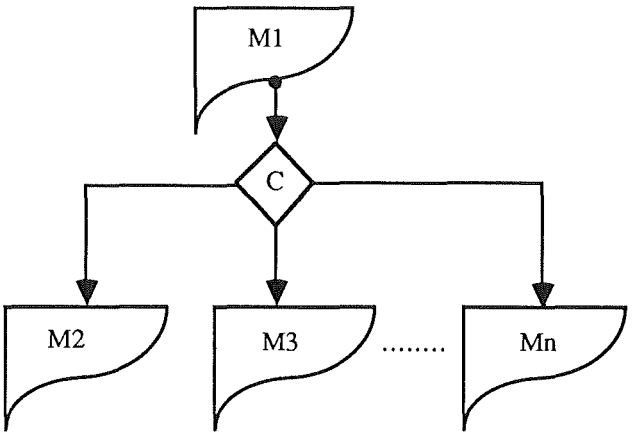


Figure IV.6. : Strucure conditionnelle

IV.1.4.3. Le schéma de la conversation

Le schéma de la conversation porte uniquement sur les messages fonctionnels interactifs car :

- Un **message d'aide** pourra virtuellement apparaître n'importe quand; l'utilisateur pourra demander ce genre d'informations à n'importe quel moment. L'expression du déclenchement du traitement d'un tel message est plus facilement représentable sous forme de précondition. La précondition énonce la condition d'occurence du message. Par exemple, il aura pour précondition la sélection de l'objet sur lequel porte le message d'aide et l'activation du menu d'aide.

- Un **message d'erreur syntaxique** sera directement lié au(x) message(s) sur le(s)quel il porte. Par exemple, un message d'erreur signalant que le numéro du client doit être un entier portera sur le message Numéro_client, source de l'erreur. Il en va de même pour un message d'erreur lié à la conversation : l'oubli de la saisie de l'adresse d'un nouveau client porte sur le message manquant[2]. La clarté du schéma de la conversation sera améliorée si l'on considère l'occurence de ce type de messages comme implicite. Sa spécification devra néanmoins contenir l'expression de son déclenchement sous forme de précondition[3].

---

[1] Dans notre exemple, l'utilisateur pourra choisir d'introduire le nom ou le numéro du client qui passe la commande.

[2] Nous verrons à la section VI.1.1.5. que l'existence d'un message d'erreur syntaxique lié à la conversation dépendra de la nature du point de synchronisation (explicite ou implicite).

[3] Voir section IV.1.5.1.

- Un **message de contrôle** peut être directement retiré du schéma de la conversation. Les structures d'enchaînement qui y sont spécifiées représentent en effet les messages de contrôle qui existeront. Par exemple, la structure parallèle constitue un message de contrôle invitant l'utilisateur à choisir parmi les alternatives qui lui sont offertes.

Le schéma de la conversation permet donc au concepteur de l'application interactive de spécifier l'enchaînement des messages interactifs fonctionnels que l'utilisateur devra respecter. Pour cela, une technique d'implémentation pourrait être l'interdiction de sélectionner les messages que l'utilisateur ne peut couramment manipuler. Les techniques qui seront utilisées sont directement liées aux objets interactifs qui présenteront les messages. Nous reportons donc le lecteur à la section IV.4.1. pour de plus amples précisions.

La spécification des messages interactifs fonctionnels, les opérations que l'utilisateur peut effectuer sur ceux-ci et le schéma de la conversation, constituent les informations nécessaires à la réalisation de la Tâche. Il nous faut à présent spécifier les informations qui concernent uniquement l'interface-utilisateur. Il s'agit des messages purement interactifs.

IV.1.5. Les messages purement interactifs

IV.1.5.1. Spécification des messages purement interactifs

Comme nous l'avions décidé lors de la spécification des messages interactifs fonctionnels, il est à présent temps de spécifier les messages purement interactifs. Rappelons qu'un message purement interactif est un message qui n'a pas de contrepartie fonctionnelle.

Comme ce fut le cas pour les messages interactifs fonctionnels, un message purement interactif sera spécifié par :

- un **nom**,
- une **définition**,
- un **type**,
- une **justification**,
- un **contenu** ou **structure de données**,
- des **contraintes sur la présentation**, et
- des **opérations de manipulation**.

Une information supplémentaire à donner sera la condition de déclenchement du traitement du message[1]. Elle sera spécifiée sous forme de précondition.

Rappelons également que nous faisons la distinction entre trois types de message purement interactif. Cette distinction a une incidence sur les opérations qu'on peut leur associer :

[1] Voir section IV.1.4.3.

IV.1.5.2. Opérations sur les messages purement interactifs

Un **message de contrôle** permet à l'utilisateur de se déplacer dans le schéma de la conversation. Voyons pour chaque structure d'enchaînement le message de contrôle qui pourra apparaître :

- *le parallélisme* : laisse le choix à l'utilisateur de l'ordre de traitement d'une liste de messages. Le message de contrôle qui pourra apparaître sera dès lors de saisie et permettra la sélection d'un message parmi l'alternative proposée (le message correspondra à l'affichage d'un menu). Nous avons vu les opérations que l'on pouvait associer à un message de saisie[1]. Notons que ce message peut ne pas exister, par exemple dans un contexte de manipulation directe affichant les différents messages et permettant à l'utilisateur de sélectionner directement celui de son choix, sans passer par un message de contrôle.

- *l'enchaînement séquentiel :* ne laisse aucune liberté à l'utilisateur. Celui-ci ne peut donc décider de son déplacement puisqu'il se fera automatiquement.

- *la structure convergente :* procède de la même manière que l'enchaînement séquentiel. Le déplacement de l'utilisateur est automatique.

- *la synchronisation :* peut être automatique ou explicite. Elle est automatique si elle est vérifiée par l'interface, sans intervention de l'utilisateur. Elle est explicite si ce dernier doit avertir l'interface de son occurence. Par exemple, une synchronisation sera automatique si la terminaison du traitement de deux messages entraîne la réalisation du point de synchronisation. Elle est explicite si le nombre de traitement d'un type de message tel que la "ligne_de_commande" est défini par l'utilisateur qui avertit l'interface lorsqu'il a terminé leur traitement. Une synchronisation explicite sera représentée par un message de saisie que l'utilisateur sélectionnera pour la déclencher.

- *L'itération de messages :* est réalisée automatiquement par l'interface. Sa terminaison a été analysée lors de la synchronisation.

- *La strucuture conditionnelle :* peut être à nouveau automatique ou explicite. Elle est automatique si elle peut être analysée par l'interface sans intervention de l'utilisateur. Elle ne nécessite alors pas de message de contrôle. Elle est explicite si l'évaluation de la condition dépend de l'utilisateur. Un message de contrôle sera alors saisi par ce dernier et analysé par l'interface pour déclencher l'enchaînement approprié[2].

En résumé, un message de contrôle sera de saisie et pourra de la sorte être manipulé par les opérations qui s'y rapportent.

Un **message d'aide** affiche à l'utilisateur, à sa demande, un texte explicatif concernant l'application en cours. Un tel message est d'affichage et n'aura donc pas de primitive liée à son contenu (cfr section II.2.2.). L'utilisateur pourra sélectionner le message pour demander de l'aide et le ranger ou le supprimer lorsqu'il en a pris connaissance. Ce sont donc les deux primitives liées à une occurence d'un message de ce type.

---

[1] Cfr section II.2.2.

[2] Par exemple le choix de la saisie du nom ou du numéro du client.

Un **message d'erreur syntaxique** a les mêmes caractéristiques qu'un message d'erreur sémantique. Puisqu'il est d'affichage, il n'existe pas de primitive liée à son contenu. Il est affiché par l'interface lorsque l'utilisateur a commis une erreur de syntaxe. Ce dernier ne peut donc pas le sélectionner mais simplement le supprimer ou le ranger.

Ceci termine les opérations sur les messages purement interactifs.

IV.2. Validation de l'analyse de la tâche par rapport à la dynamique des traitements

Les messages fonctionnels interactifs proviennent d'une décomposition ou d'un regroupement des messages fonctionnels. Il est donc nécessaire de vérifier que toutes les informations se retrouvant dans les messages fonctionnels apparaissent également dans les messages interactifs fonctionnels.

C'est le but de cette phase. Elle consiste en la vérification de cette correspondance 1 à 1 (ni redondance, ni incomplétude) entre les deux types de message. Il faudra vérifier que chaque attribut se trouve dans un et un seul message.

IV.3. Dynamique d'implémentation

Nous avons, lors de l'analyse fonctionnelle, spécifié l'enchaînement des traitements et des messages fonctionnels. Ensuite, nous avons spécifié le schéma de la conversation c'est-à-dire l'enchaînement des messages fonctionnels interactifs pour enfin spécifier les messages purement interactifs.

A présent, nous allons spécifier l'ordre exact dans lequel se dérouleront en réalité les traitements. Un traitement signifie ici une fonction au sens de la méthodologie proposée par Bodart et Pigneur. *"Une fonction correspond au niveau élémentaire de la nomenclature des traitements; elle résulte de la décomposition d'une phase en sous-traitements. Elle est associée à un objectif et un comportement considérés comme élémentaires par l'organisation, l'utilisateur et le concepteur du système d'information"* [Bodart & Pigneur, 1988]. Pour ce faire nous allons greffer au schéma de la conversation les traitements résultant de l'analyse fonctionnelle. Le résultat sera appelé le **schéma de la dynamique d'implémentation**.

Ce schéma permet de fixer de manière précise l'enchaînement des traitements en fonction de la conversation. Il pourra être présenté à l'utilisateur dans le but de mieux lui faire comprendre les principes fonctionnels de l'application. Il sera également indispensable au concepteur de l'interface puisque la réalisation de l'interface devra respecter ce schéma.

La dynamique d'implémentation respectera la notation utilisée pour la conversation. Cela ne devrait en principe pas poser de problèmes majeurs pour l'insertion des traitements car la notation utilisée pour la conversation devra constituer une extension compatible à la dynamique des traitements.

Nous en avons terminé de l'analyse de la Tâche et de son intégration aux fonctionnalités. Il nous reste à spécifier la Présentation qui sera associée aux messages interactifs.

IV.4. Spécification de la Présentation

IV.4.1. Introduction

Nous devons maintenant définir les caractéristiques des objets qui représenteront les messages interactifs que nous avons spécifiés.

Un objet interactif est la visualisation à l'écran d'un ou plusieurs messages interactifs. Notons également qu'un même message interactif pourra se voir représenté par plus d'un objet interactif.

Un type d'objet interactif est dépendant de l'environnement dans lequel se déroulera l'application. Par exemple, une fenêtre sera constituée d'éléments constitutifs différents selon l'ordinateur ou le système qui sera utilisé. De même, il existe différents types de fenêtre offert par le même type d'environnement. Il est donc important de spécifier les caractéristiques d'un type d'objet interactif et les opérations qu'on pourra lui associer.

Nous parlons ici de **type** ou **classe** d'objet interactif; les objets interactifs que nous spécifierons seront des occurences d'un tel type.

Le concepteur d'une application interactive dispose d'objets standard, offerts par le système sur lequel sera implémentée l'application interactive mais il peut aussi en créer d'autres. Pour l'un ou l'autre de ces types d'objet, il est important de fournir à l'utilisateur un manuel d'utilisation. Par exemple, l'utilisateur doit savoir que s'il clique sur le bouton de fermeture d'une fenêtre, il provoquera la disparition de la fenêtre de l'écran et que cela pourra avoir des répercussions sur l'application en cours. La spécification des objets interactifs qui suit constituera la base de la construction de ce manuel.

IV.4.2. Spécification d'un objet interactif

Un objet interactif sera spécifié à l'aide des informations suivantes :

- son **Nom** : Nom de l'objet interactif;

- son **Type** : Type de l'objet interactif dont il est une occurence. Rappelons qu'un objet interactif pourra être standard, créé spécialement pour l'application ou encore faire partie d'une librairie d'objets créés par le concepteur;

- sa **Justification** : Explicitation de la raison pour laquelle on a choisi ce type d'objet interactif. Tout comme il était important de justifier la découpe des messages interactifs fonctionnels, il est important de bien choisir la représentation qui en sera faite. Le succès du Macintosh est dû en grande partie à la métaphore qu'il utilise pour présenter les objets que manipule l'utilisateur comme des éléments d'un bureau qu'il peut ranger, jeter dans la

corbeille... Une nouvelle fois, le concepteur devra connaître l'utilisateur pour pouvoir choisir les objets interactifs qui correspondront le mieux à son mode de pensée.

- son **Contenu ou Visualisation** : Visualisation de l'objet à l'écran. Ceci peut se faire à l'aide d'un dessin de l'objet interactif ou d'un texte explicatif;

- le(s) **Message(s) interactif(s) représenté(s)** : Le ou les messages interactifs auxquels l'objet correspond. Ceci permet la vérification rapide des objets interactifs de sorte que tous les messages interactifs soient effectivement présentés à l'utilisateur.

- les **Opérations portant sur le contenu sémantique ou la visualisation** : Nous avons vu que, dans la spécification des messages interactifs, on définissait les opérations de manipulation liées au message interactif. On a également parlé des actions qu'on pouvait associer à un type d'objet interactif. Il faudra à présent faire le lien entre les opérations que l'utilisateur peut effectuer sur l'objet interactif et les actions de manipulation de l'objet qui les représenteront.

  Les actions de manipulation d'un objet n'ont pas toutes un équivalent sémantique. Par exemple, l'action qui consiste à réduire la taille d'une fenêtre ne concerne que l'interface. Il n'est pas utile de spécifier ce type d'action de manipulation car on peut le considérer comme intrinsèque au type d'objet. Ces actions font en effet partie de la définition de l'objet et n'apporteraient pas d'information supplémentaire dans les spécifications. Par contre, elles les alourdiraient inutilement.

  En résumé, nous reprendrons chaque opération sémantique définie lors de la spécification du message interactif représenté par l'objet en cours de spécification, et nous y associerons l'action de manipulation de l'objet qui y correspond.

- **les contraintes d'enchaînement** : Elles découlent directement des contraintes d'enchaînement définies lors de la spécification des messages interactifs. Les actions physiques étant la traduction des opérations que l'utilisateur pourra effectuer sur les messages, elles devront respecter les mêmes enchaînements. On ne fait que les reprendre pour plus de clarté.

- **Les fonctions déclenchées** : Certaines actions sur les objets interactifs déclencheront une ou plusieurs fonctions de l'application proprement dite (rappelons que ces fonctions découlent de l'analyse fonctionnelle où une phase est décomposée en fonctions selon des critères de sémantique simple, de comportement organisationnel élémentaire...). Il nous parait utile de les reprendre ici bien qu'il soit possible de les déduire du schéma de la dynamique d'implémentation puisqu'un objet représente un ou plusieurs messages interactifs ou parties de messages interactifs qui déclenchent certaines fonctions.

IV.5. Conclusion

Nous avons à présent tous les éléments nécessaires à la spécification de l'interface-utilisateur d'une application interactive. Cependant, tout comme le design d'une application interactive, une méthodologie se doit d'être testée et évaluée de manière à pouvoir être affinée. C'est ce que nous ferons au chapitre suivant.

La méthodologie que nous proposons, si elle doit permettre au concepteur de spécifier toutes les informations dont il a besoin pour la réalisation de son application interactive, nécessite encore une grande part de réflexion de sa part. Nous avons vu au premier chapitre qu'il devait connaître l'utilisateur pour lequel l'application sera réalisée. Cette connaissance se reflètera dans la découpe en messages interactifs, dans la spécification du schéma de la conversation ainsi que dans la présentation des messages interactifs.

Notre méthodologie ne prétend pas supprimer le besoin d'itération dans la conception d'application interactive mais de le diminuer en mettant à la disposition du concepteur un environnement de spécification qui pourra être validé par l'utilisateur et cela avant même l'implémentation de l'application interactive. L'idéal serait bien sûr de disposer d'un outil de spécification implémentant la méthodologie et permettant la conception d'un prototype de l'interface-utilisateur spécifiée. Cette possibilité sera analysée au chapitre VII.

Nous rappelons que nous n'en sommes encore qu'à la première version de la méthodologie. Il est nécessaire de la tester en suivant pas à pas les étapes que nous avons spécifiées et en tenant un journal de bord sur lequel nous noterons toutes les remarques qui peuvent être faites. C'est ce que nous proposons au chapitre suivant.

Chapitre 5
Cadre d'expérimentation

V.0. Introduction

La méthodologie que nous avons présentée au chapitre précédent n'en est qu'à sa phase d'élaboration. Dans le but de la tester et ensuite de l'améliorer, il est nécessaire de l'appliquer à la réalisation d'une application interactive. Toutes les remarques qui peuvent être faites pendant la conception doivent alors être soigneusement notées pour en faire l'analyse ultérieure.

Ce chapitre présente les spécifications de l'application interactive que nous réaliserons pour tester notre méthodologie. Nous débuterons tout d'abord par une présentation du problème et les hypothèses que nous avons posées pour le résoudre. Nous spécifierons ensuite les fonctionnalités de l'application interactive selon la méthode proposée par F. Bodart et Y. Pigneur (1983).

V.1. Présentation et hypothèses

L'expérimentation que nous conduirons au chapitre suivant a pour but de réaliser une application interactive de saisie d'une commande client. Il s'agit d'une phase bien précise qui se retrouve dans toute entreprise commerciale de vente et qui fera généralement partie d'un projet plus vaste de gestion des clients, de leurs commandes, des produits que l'entreprise vend... Nous nous concentrerons uniquement sur la saisie de la commande que passe un client pour l'achat de certains produits vendus par la firme.

On peut concevoir deux manières par lesquelles le client peut passer une commande : soit il téléphone à la société et passe sa commande en dialoguant avec l'opératrice, soit il remplit le bon de commande qui par exemple accompagne le catalogue dont il dispose et l'envoie à l'entreprise.

Ces deux façons qu'a le client de commander les produits qu'il désire sont différentes à plusieurs égards : le facteur temps peut être considéré comme primordial lors de la saisie d'une commande téléphonée; le client aura fait l'effort de communiquer avec l'entreprise et voudra de plus connaître immédiatement si sa commande peut être livrée sans délai c'est-à-dire si les produits qu'il souhaite sont en stock. L'opératrice doit pouvoir vérifier si toutes les informations que lui communique le client sont valides (son numéro de client, les numéros de produit commandés...) En résumé, le client doit savoir à la fin de la communication tout ce qui l'intéresse au sujet de sa commande; il ne devra plus téléphoner une seconde fois pour enfin avoir les renseignements qu'il désirait (à moins, bien sûr, qu'il n'en ait pas parlé au cours de son premier entretien).

Par contre, le facteur temps dans le transfert d'un bon de commande dans le système d'information que possède l'entreprise n'est important qu'en tant que critère d'efficacité de la personne qui effectuera le travail. Les informations que devra fournir l'application interactive seront davantage destinées à l'utilisateur qu'au client puisque celui-ci sera absent de l'interaction. Un bon de commande pourra être incomplet ou invalide, ce qui nécessitera son renvoi à l'expéditeur.

De ce qui précède nous pouvons tirer plusieurs conclusions sur l'application interactive qui devra être réalisée :

- Il existera **deux interfaces** pour l'application interactive : l'une pour la saisie de commandes téléphonées, l'autre pour la saisie de bons de commande. En effet, nous avons vu que l'interface ne devait pas se comporter de la même manière suivant le type de saisie. Une commande téléphonée devra avant tout fournir le plus rapidement possible un maximum d'informations pour que l'utilisateur puisse les communiquer au client. Celui-ci pourra modifier sa commande en fonction des éléments qui lui sont fournis. Par exemple, le fait de savoir qu'il devra attendre deux mois avant de recevoir les deux polos gris qu'il voulait commander parce qu'il y a une rupture de stock le fera peut-être changer d'avis et commander des polos roses. De plus, les validations opérées par les fonctionnalités ou l'interface devront être répercutées immédiatement à l'utilisateur pour qu'une nouvelle fois il en informe le client.

La saisie d'un bon de commande pourra se faire de façon plus souple et en respectant également mieux le mode de pensée de l'utilisateur. Celui-ci n'aura plus le client pour en quelque sorte lui imposer la marche à suivre et ne sera également pas pressé de la même manière par le temps.

- Il y aura **deux classes d'utilisateur** : la saisie d'une commande téléphonée ou manuscrite ne nécessite pas les mêmes compétences de la part de l'utilisateur. Nous avons déjà relevé les différences qui existaient dans l'accomplissement de ces deux types de saisie. L'utilisateur qui devra communiquer avec le client devra avoir un profil psychologique d'ouverture à autrui, ses capacités physiques et mentales devront lui permettre d'effectuer rapidement la saisie des instructions du client... La réalisation des deux interfaces devra donc tenir compte de la classe d'utilisateur avec laquelle l'interface communiquera.

- Il est important de souligner que les deux interfaces partageront **les mêmes fonctionnalités** : les opérations sémantiques effectuant la saisie d'une commande ne varient pas selon le type d'interface. Les fonctionnalités ne manipulent en effet que des informations sémantiques, communes à toute interface.

Les fonctionnalités de l'application interactive à réaliser consistent donc en la validation et l'enregistrement d'une commande. L'utilisateur entrera le numéro du client qui passe la commande ou son nom s'il s'agit d'un nouveau client, l'adresse du client si celui-ci a changé d'adresse ou s'il est nouveau, les lignes de la commande constituées d'un numéro de produit et d'une quantité. La commande sera enregistrée si toutes les informations qui la constituent ont été validées.

Une définition plus précise des fonctionnalités est donnée à la section suivante.

V.2. Analyse fonctionnelle

V.2.1. La phase Enregistrement d'une commande-client

a pour objectif de vérifier une commande-client à enregistrer et, lorsqu'elle est valide, de la mémoriser. De plus, elle doit permettre de mémoriser un client qui commande pour la première fois et d'enregistrer un changement d'adresse communiqué par un ancien client;

est effectuée dans le service de réception des commandes

génère une commande enregistrée, si la mémorisation a été possible

utilise et/ou modifie une mémoire qui comprend :
- des commandes, leur provenance et leurs lignes;
- des clients;
- des produits et leurs substitutions;
- des lignes de produits

reçoit une commande à enregistrer qui peut comprendre :
- un numéro de client,
- un nom de client,
- une adresse de domicile,
- des numéros de produit et des quantités commandées

intègre les fonctions suivantes :

- mémorisation d'une commande,
- mémorisation d'un client,
- modification de l'adresse d'un client,
- validation d'une commande,
- validation d'une provenance,
- validation d'une ligne,
- validation d'un client et
- validation d'un produit.

V.2.2. Structure de données et contraintes d'intégrité

La mémoire de la phase Enregistrement d'une commande-client est représentée par les ensembles suivants :

- l'ensemble des entités CLIENT(S),
  caractérisées par
  un Numéro de client,
  un Nom de client,
  une Adresse de domicile,
  une Limite d'achat et
  une Catégorie;

- l'ensemble des entités PRODUIT(S),
  caractérisées par
  un Numéro de produit,
  un Libellé de produit,
  une Unité d'achat,
  un Prix unitaire d'achat,
  un Etat d'approvisionnement et
  une Date de réapprovisionnement;

- l'ensemble des associations SUBSTITUTION(s),
  reliant un Produit (Remplacé) et son Produit (Analogue);

- l'ensemble des entités COMMANDE(s),
  caractérisées par
  un Numéro de commande,
  une Date de commande,
  un Montant total de commande et
  un Rabais de commande;

- l'ensemble des associations PROVENANCE(s),
  reliant une Commande (Emise) à son Client (Emetteur);

- l'ensemble des associations LIGNES(s),
  reliant une Commande (Conernée) à un Produit (Commandé), et caractérisé par
  une Quantité commandée et
  un Montant de ligne;

Les données véhiculées dans la phase Enregistrement d'une commande-client sont représentées par les messages suivants :

- Commande enregistrée,
- Commande valide,
- Provenance valide,
- Ligne valide,
- Client valide,
- Produit valide,
- Commande à enregistrer.

Une entité PRODUIT

représente un article figurant au dernier catalogue diffusé par la firme;

est caractérisé par :

- un Numéro de produit
  référence de l'article telle qu'elle apparaît dans le catalogue (les 2 premiers chiffres représentent le "rayon d'achat", les 5 chiffres suivants représentent l'article taille/couleur" et le dernier chiffre représente le "check digit" - le reste de la division entière des 7 premiers chiffres par 7),

- un Libellé de produit
  descriptif de l'article, repris dans le catalogue,
- une Unité d'achat
  nombre d'articles faisant l'objet d'un lot non partionnable,
- un Etat d'approvisionnement
  indicateur signalant la situation du produit en stock
  (peut être 'en stock', 'en avis d'attente' ou 'épuisé') et
- une Date de réapprovisionnement (facultative)
  indication sur la date probable de réassortiment (pour un Produit dont l'Etat d'approvisionnement est "en avis d'attente");

joue le rôle de :
- Produit (Commandé) dans aucune ou plusieurs Lignes(s),
- Produit (Analogue) dans aucune ou plusieurs Substitution(s) et
- Produit (Remplacé) dans aucune ou une seule Substitution;

est identifié par son Numéro de produit.

Une association SUBSTITUTION

représente l'existence, pour un produit épuisé, d'un autre produit analogue qui peut lui être substitué dans toute commande;

relie un Produit (Remplacé)
(dont l'Etat d'approvisionnement est 'épuisé') et
son Produit (Analogue)
(dont l'Etat d'approvisionnement n'est pas 'épuisé');

est identifié par le Produit (Remplacé).

Une entité COMMANDE

représente un ordre de commande passé par un client auprès de la société pour un ou plusieurs produits;

est caractérisé par :

- un Numéro de commande
  numéro attribué par compostage lors de l'enregistrement de la Commande
- une Date de commande
  date du jour où la commande a été enregistrée,
- un Montant total de commande
  somme des Montant(s) de ligne pour toutes les Ligne(s) de la Commande (Concernée) et
- un Rabais de commande (facultatif)
  réduction de 20% du Montant total de la commande
  (accordé à un client (Emetteur) dont la catégorie est 'collaborateur');

joue le rôle de
- Commande (Emise) dans une et une seule Provenance et
- Commande (Concernée) dans une ou plusieurs Ligne(s);

est identifiée par son Numéro de commande;

[ne peut être ajoutée que si le Montant total de la commande, réduit du Rabais de commande éventuel, est inférieur à la Limite d'achat du Client (Emetteur)].

Une association PROVENANCE

représente l'engagement contractuel entre un client et la société pour une commande donnée;

relie une Commande (Emise) et
son Client (Emetteur) [dont la Catégorie n'est pas 'douteux'];

est identifiée par la Commande (Emise).

Une association LIGNE

représente la promesse de livraison d'un article commandé;

relie une Commande (Concernée) et
un Produit (Commandé)
[dont l'Etat d'approvisionnement n'est pas 'épuisé'];

est caractérisé par :
- une Quantité commandée
nombre d'Unités d'achat commandées du Produit (Commandé) et
- un Montant de ligne
produit de la Quantité commandée par le Prix unitaire d'achat du Produit (Commandé);

est identifié par la Commande (Concernée) et le Produit (Commandé).

Une entité CLIENT

représente une personne, physique ou morale, appartenant au marché des acheteurs potentiels ou réels de la société;

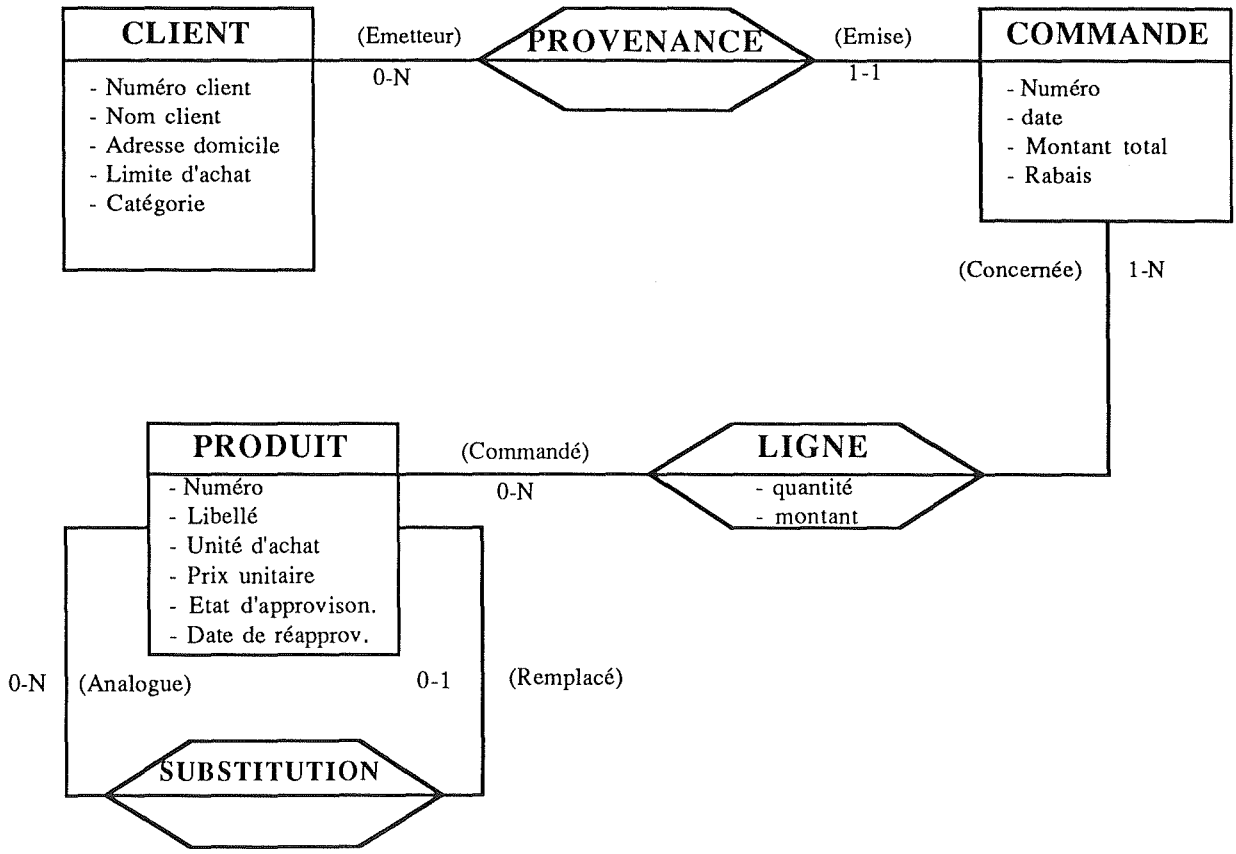
est caractérisé par :
- un Numéro de client
numéro attribué par compostage à l'enregistrement d'un nouveau client,
- un Nom de client (Nom, Prénom, Titre)
renseignement fournis par le client lors de son premier contact,
- une Adresse de domicile (Rue, Numéro, NPA, Localité)
dernier domicilie connu, donné par le client,

- une Limite d'achat
  montant du crédit maximal attribué au client
  [est initialement de FB 2500, à l'enregistrement d'un 'nouveau' client] et
- une Catégorie
  indication sur l'état des relations entre le client et la société:
  ['régulier' s'il a acheté dans les 12 derniers mois,
  'nouveau' s'il s'agit de son premier contact avec la société,
  'ancien' s'il n'a plus acheté depuis 12 mois,
  'douteux' s'il est repris sur la "liste noire" de la société, et
  'collaborateur' s'il est un employé de la société];

joue le rôle de Client (Emetteur) dans aucune ou plusieurs Provenance(s);

est identifié par son Numéro de client.

V.2.3. Schéma entités/associations



.C.Figure 5.1. : Schéma entité/association

V.2.4. Messages fonctionnels

- En sortie de la phase :

Un message COMMANDE ENREGISTREE
comprend une Commande, sa Provenance et ses Ligne(s);

- interne à la phase :

Un message COMMANDE VALIDE
comprend une Provenance valide,
des Ligne(s) valide(s), au moins une,
[qui portent sur des produits différents],
un Montant total de commande
[somme des Montant(s) de ligne des Ligne(s) valide(s)]
un Rabais de commande (facultatif)
[si la Catégorie du Client valide est 'collaborateur'];

Un message PROVENANCE VALIDE
comprend un Client valide
[dont la Catégorie n'est pas 'douteux'];

Un message LIGNE VALIDE
comprend un Produit valide
[dont l'Etat d'approvisionnement n'est pas 'épuisé'],
une Quantité commandée et
un Montant de ligne
[produit de la Quantité commandée par le Prix unitaire d'achat du Produit valide];

Un message CLIENT VALIDE
comprend un Client
[qui existe dans la mémoire];

Un message PRODUIT VALIDE
comprend un Produit
[qui existe dans la mémoire];

- en entrée de la phase :

Un message COMMANDE A ENREGISTRER
peut comprendre :
un Numéro de client,
un Nom de client (Nom, Prénom, Titre),
une Adresse de domicile (Rue, Numéro, NPA, Localité),
des Numéro(s) de produit et
des Quantité(s) commandée(s).

V.2.5. Description des traitements

Les traitements de la phase Enregistrement d'une commande-client sont représentés par les fonctions suivantes :

* la fonction MEMORISATION D'UNE COMMANDE
  génère une Commande enregistrée
  ajoute une COMMANDE,
    une PROVENANCE et
    des LIGNE(S), au moins une
  modifie la Limite d'achat du Client (Emetteur)
  reçoit une Commande valide

- la fonction VALIDATION D'UNE COMMANDE
  génère une Commande valide, si possible
  reçoit une Provenance valide et
    des Ligne(s) valide(s), mais au moins une

- la fonction VALIDATION D'UNE PROVENANCE
  génère une Provenance valide, si possible
  reçoit un Client valide

- la fonction VALIDATION D'UNE LIGNE
  génère une Ligne valide, si possible
  reçoit un Produit valide
    une Quantité commandée (facultative)

* la fonction MODIFICATION DE L'ADRESSE D'UN CLIENT
  modifie l' Adresse de domicile du Client
  reçoit un Client valide et
    une Adresse de domicile

* la fonction MEMORISATION D'UN CLIENT
  génère un Client valide;
  ajoute un CLIENT
  reçoit un Nom de client et
    un Adresse de domicile

- la fonction VALIDATION D'UN CLIENT
  génère un Client valide, si possible
  reçoit un Numéro de client

- la fonction VALIDATION D'UN PRODUIT
  génère un Produit valide, si possible
  reçoit un Numéro de produit

La fonction MEMORISATION D'UNE COMMANDE

a pour objectif de créer et de mémoriser une nouvelle commande-client valide, avec sa provenance et ses lignes de commande;

génère une Commande enregistrée

ajoute une COMMANDE
[dont le Numéro de commande est attribué par compostage
la Date de commande est celle du jour
le Montant total de commande est celui de la Commande valide
le Rabais de commande éventuel est celui de la Commande valide];

modifie la Limite d'achat du Client (Expéditeur)
[en lui soustrayant le Montant total de commande, réduit du Rabais de commande éventuel de la Commande valide];

reçoit une Commande valide.

La fonction VALIDATION D'UNE COMMANDE

a pour objectif de garantir, à partir des lignes de commande et du client, que :
- les lignes portent sur des produits différents,
- le montant de la commande est inférieur à la limite d'achat du client
- le rabais de commande est établi s'il s'agit d'un employé de la société;

génère une Commande valide
si [le Montant total de commande, réduit du Rabais de commande éventuel, est inférieur à la Limite d'achat du Client Valide];

applique les règles :

1. si plusieur Ligne(s) valide(s) portent sur un même produit,
alors il faut les remplacer par une seule Ligne Valide dont :
. la Quantité commandée est la somme des Quantité(s) commandée(s) initiales
. le Montant de ligne est la somme des Montant(s) de ligne initiaux

2. le Montant total de commande est la somme des Montant(s) de ligne pour toutes les Ligne(s) valide(s), après application de la règle 1

3. si la Catégorie du Client valide est 'collaborateur',
alors le Rabais de commande est établi (20% du Montant total de commande).

reçoit une Provenance valide et
des Ligne(s) valide(s), mais ou moins une;

La fonction VALIDATION D'UNE PROVENANCE

a pour objectif de garantir que le client n'est pas sur la 'liste noire';

génère une Provenance valide
si [la Catégorie du Client valide n'est pas 'douteux'];

reçoit un Client valide.

La fonction VALIDATION D'UNE LIGNE

a pour objectif de garantir que la ligne porte sur un produit non épuisé et que, pour un tel produit, la quantité commandée est fixée;

génère une Ligne valide
si [l'Etat d'approvisionnment du Produit valide n'est pas 'épuisé'];

applique les règles :

1. si l'Etat d'approvisionnement du Produit valide est 'épuisé' et si celui-ci peut être substitué par un Produit (Analogue) alors il faut remplacer le Produit valide par son Produit (Analogue)

2. si la Quantité commandée n'est pas explicitement fournier, alors elle est d'UNE unité d'achat

3. le Montant de ligne est le produit de la Quantité commandée par le Prix unitaire d'achat du Produit, après application des règles 1 et 2;

reçoit un Produit valide et
une Quantité commandée (facultative).

La fonction MODIFICATION DE L'ADRESSE D'UN CLIENT

a pour objectif d'enregistrer un changement d'adresse communiqué par un client, qui existe déjà dans la mémoire;

modifie l'Adresse de domicile du CLIENT
[en lui substituant l'Adresse de (nouveau) domicile fournie];

reçoit un Client valide et
une Adresse de (nouveau) domicile (Rue, Numéro, NPA, Localité).

La fonction MEMORISATION D'UN CLIENT

a pour objectif de créer et de mémoriser un nouveau client;

génère un Client valide;

ajoute un CLIENT
[dont le Numéro de client est attribué par compostage
le Nom de client est celui donné en entrée
l'Adresse de domicile est celle donnée en entrée
la Limite d'achat est de 2500 FB
la Catégorie est 'nouveau']

reçoit un Nom de client (Nom, Prénom, Titre) et
une Adresse de domicile (Rue, Numéro, NPA, Localité).

La fonction VALIDATION D'UN CLIENT

a pour objectif d'identifier un client, existant dans la mémoire, à partir de son seul numéro de client;

génère un Client valide
si [il existe dans la mémoire un client avec ce Numéro de client];

reçoit un Numéro de client.

La fonction VALIDATION D'UN PRODUIT

a pour objectif d'identifier un produit, existant dans la mémoire, à partir de son seul numéro de produit;

génère un Produit valide
si [il existe dans la mémoire un Produit avec ce Numéro de produit];

reçoit un Numéro de produit.

V.3. Dynamique des traitements

Le schéma de la dynamique des traitements est présenté à la figure V.2. Cinq types de message fonctionnel sont nécessaires à l'exécution des fonctions de l'application : Numéro_client, Nom_et_adresse_client, Adresse_domicile, Numéro_de_produit et Quantité_commandée. Il y aura en réalité un seul des messages Numéro_client et Nom_et_adresse_client qui sera traité, selon qu'il s'agit d'un ancien ou d'un nouveau client. La saisie du Numéro_client provoquera la validation du numéro (il doit correspondre à un client existant) et de la provenance (le client ne peut pas être "douteux"). L'adresse_domicile du client pourra être modifiée en cas de déménagement. L'occurence du message Nom_et_adresse_client déclenchera la fonction d'enregistrement du nouveau client.

Le message Numéro_de_produit devra correspondre à un produit existant. La Quantité_commandée ne devra pas dépasser la quantité disponible du produit commandé.

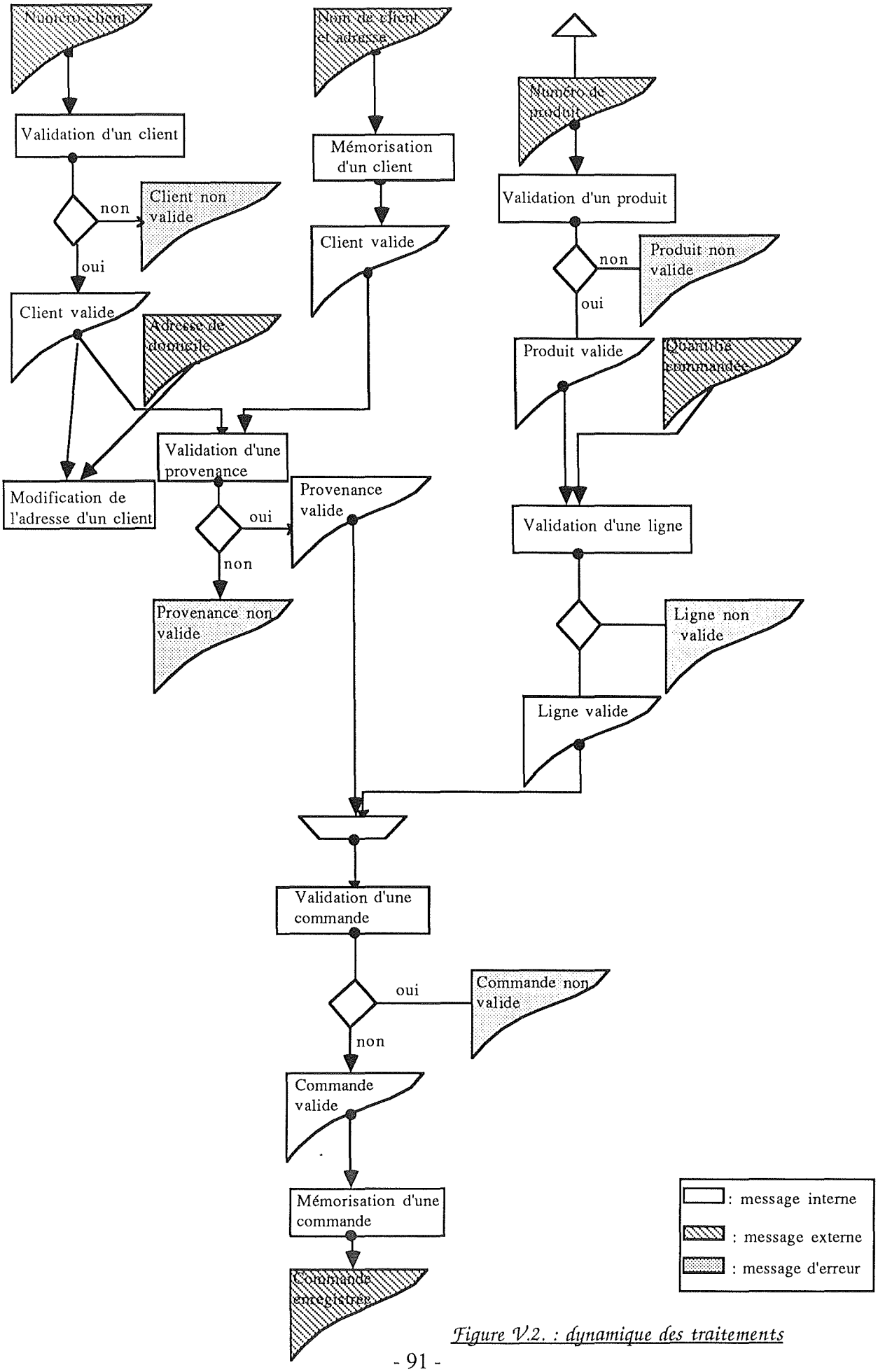


Figure V.2. : dynamique des traitements

Si l'occurence du message correspondant aux références du client et un nombre quelconque de lignes de commande (mais au moins une) ont pu être validées, la commande sera enregistrée si son montant ne dépasse pas la limite d'achat du client.

V.4. Conclusion

Nous avons dans ce chapitre présenté et spécifié les fonctionnalités de l'application interactive sur laquelle nous allions nous baser pour tester notre méthodologie.

Les informations que nous avons définies nous permettent de connaître les besoins de l'application proprement dite afin de réaliser ses traitements. Nous connaissons également l'objectif des différentes fonctions de l'application interactive. A partir de ces renseignements, nous pouvons spécifier la composante de notre architecture qui dialoguera avec l'opérateur. Rappelons qu'il est également indispensable de procéder à une étude des caractéristiques de ce dernier pour y parvenir correctement. Nous concevrons deux interfaces qui correspondront à la saisie d'une commande téléphonée et à partir d'un bon de commande.

Le chapitre suivant examinera les réalisations qui ont été produites.

Chapitre VI
Application et évaluation de la méthodologie

VI.0. Introduction

Nous avons développé au cours du chapitre IV tous les éléments qui devaient permettre à un concepteur de spécifier une ou plusieurs interfaces d'une application interactive. Il nous faut à présent les évaluer de manière à améliorer la qualité de la méthodologie.

Ce chapitre respectera l'ordre de spécification qui a été décrit au chapitre IV. Nous débuterons par la spécification de la Tâche; elle se fera une fois pour toute par interface car elle est indépendante de toute implémentation. Nous la décrirons pour une commande écrite et téléphonée. Nous l'évaluerons ensuite.

Nous passerons alors à la validation de la Tâche par rapport à la dynamique des traitements et au schéma de la dynamique d'implémentation.

La spécification de la Présentation dépend de l'implémentation qui sera faite de l'application interactive. Nous utiliserons deux logiciels différents pour la réaliser. Il s'agira d'Hypercard et Turbo/Pascal. Ils seront tous les deux présentés brièvement. Nous spécifierons alors la Présentation qui leur est propre pour ensuite évaluer les résultats de leurs spécifications[1]. Nous terminerons par une évaluation générale de la méthodologie.

L'approche suivie dans ce chapitre est la suivante : nous exposerons dans un premier temps les spécifications de notre exemple; nous énoncerons alors toutes les remarques que nous avons pu faire lors de leur conception. Elles constituent le carnet de bord que nous avons élaboré au fur et à mesure que nous spécifions notre exemple. Enfin nous les évaluerons par rapport aux hyptohèses que nous avions élaborées lors de la création de la méthodologie. Nous reprenons ces dernières dans la section suivante.

VI.1. Hypothèses

Nous avons posé un certain nombre d'hypothèses lors de la présentation de notre méthodologie. Nous les résumons ici pour les soumettre ensuite aux résultats de la spécification de notre exemple :

- **Indépendance de conception entre les fonctionnalités et l'interface** : La séparation modulaire entre les fonctions et l'interface[2] permet de spécifier plusieurs interfaces pour une même application. Les fonctions doivent alors être indépendantes de toute interface réalisée.

- **Spécification des fonctionnalités avant l'interface** : Nous avons signalé à la section I.5.1.1. que notre méthodologie utilisait l'approche de spécification la plus ancienne : les fonctionnalités était définies avant l'interface. Cette dernière était un module d'entrée-sortie destinée à fournir aux fonctions les informations dont elles avaient besoin pour s'exécuter.

---

[1] Les spécifications réalisées pour Turbo/Pascal sont exposées à l'annexe 2.

[2] Cfr section III.1.

- **Spécification de la Tâche indépendante de la Présentation** : La Tâche concerne les informations que manipulera l'interface, sans présumer de l'implémentation (les objets interactifs) qui en sera faite.

- **La Tâche permet la définition précise de l'interface** : les informations qui seront définies lors de cette phase seront suffisantes pour créer l'entièreté du dialogue, sans être trop complexes à élaborer.

VI.2. Spécification de la Tâche

VI.1.0. Introduction

L'analyse de la Tâche se fait indépendamment de toute implémentation car elle porte sur les informations qui seront communiquées à l'utilisateur et non sur leur présentation[1].

Nous spécifierons dès lors les messages interactifs de la Tâche, les opérations que nous pouvons leur associer et le schéma de la conversation une seule fois par interface, quelque soit son ou ses implémentations futures. Rappelons que nous offrirons à l'utilisateur la possibilité d'enregistrer une commande téléphonée ou à partir d'un bon de commande. Nous débutons par la spécification de l'interface pour une commande écrite.

VI.1.1. Spécification de la tâche pour un bon de commande

VI.1.1.1. Spécification des messages interactifs fonctionnels

Les informations contenues dans les messages interactifs fonctionnels proviennent des messages fonctionnels. Leur regroupement doit non plus se faire en fonction des traitements mais de l'utilisateur. Nous présentons les messages interactifs fonctionnels que nous avons spécifiés pour cette interface :

Num_client_à_saisir

Nom : num_client_à_saisir;
Définition : un numéro_client est un numéro attribué par compostage lors de la création du client. Il est identifiant;
Type : message de saisie ;
-- ce message sera saisi lorsque l'utilisateur entrera les coordonées d'un client existant;
Justification : ce message représente l'information que l'opérateur doit introduire pour spécifier les coordonnées du client qui passe la commande[2];
Structure de données : num_client : entier;
Contraintes d'intégrité : num_client doit être un entier;
Contraintes sur la présentation : le message devra apparaître au même endroit et être présenté de la même manière que sur le bon de commande[3];

---

[1] C'est notre troisième hypothèse.

[2] Le message interactif fonctionnel correspond au message fonctionnel car on peut le considérer comme valide d'un point de vue psychologique ou ergonomique. Voir section suivante.

[3] La présentation de l'application interactive sera semblable au bon de commande que saisit l'utilisateur. Voir VI.1.1.2.

Opérations de manipulation :

- l'utilisateur peut sélectionner et clôturer le message, affecter une valeur à son attribut, la corriger ou la supprimer;
- s'il ne saisit pas ce message, il devra alors saisir le message Nom_client;

Nom_client_à_saisir

Nom : nom_client_à_saisir;
Définition : un nom_client est composé du nom du client, son prénom et son titre;
Type : message de saisie;
-- le message est de saisie si la personne qui passe la commande est un nouveau client;
Justification : ce message comprend les informations nécessaires à l'introduction d'un nouveau client;
Structure de données :

. nom : char[40];
. prénom : char[40];
. titre : char[40];

Contraintes sur la présentation : le message devra être visualisé au même endroit et être présenté de la même manière que sur le bon de commande;
Opérations de manipulation :

- l'utilisateur peut (si le client est nouveau) sélectionner et clôturer ce message, affecter une valeur à l'un de ses attributs, la corriger ou la supprimer;
- s'il ne saisit pas ce message, il devra alors saisir le message Num_client;

Adr_domicile_à_saisir

Nom : adr_domicile_à_saisir;
Définition : le message représente le domicile du client. On y acheminera les marchandises commandées;
Type : message de saisie;
-- le message sera saisi par l'utilisateur si le client est nouveau ou s'il a changé d'adresse;
Justification : ce message représente l'endroit où devront être acheminés les commandes et l'endroit où réside le client;
Structure de données :

. rue : char[40];
. numéro : char[6];
. NPA : char[6];
. localité : char[40];

Contraintes d'intégrité :

- le NPA doit être un entier positif;

Contraintes sur la présentation : l'adresse devra figurer sur deux lignes séparées avec une première pour la rue et le numéro et une deuxième pour le code postal et la localité (comme pour le bon de commande);
Opérations de manipulation :

- l'utilisateur peut sélectionner et clôturer le message, affecter une valeur à l'un de ses attributs, la corriger et la supprimer s'il a préalablement saisi le message Num_client_à_saisir;
- il doit affecter une valeur aux attributs s'il a saisi le message Nom_client_à_saisir;

Ligne_de_commande

Nom : ligne_de_cmde;
Définition : le message représente la quantité d'un produit que le client souhaite commander;
Type : message de saisie;
Justification : Une ligne de commande représente les informations nécessaires à l'envoi d'un produit en une certaine quantité. Il constitue une ligne du bon de commande. La quantité à commander est directement liée au produit sur lequel elle porte et la réunion de ces deux informations représente une unité pour l'utilisateur;
Structure de données :

. num_produit : entier;
. quantité : entier;

Contraintes d'intégrité :

- le numéro de produit doit être un entier positif dont la division des 7 premiers chiffres par 7 doit être égale au huitième;
- la quantité doit être entière positive;
- il doit y avoir au minimum une ligne de commande valide

Contraintes sur la présentation : les messages Lignes_de_commande devront être visualisés à l'utilisateur de la même manière que les lignes de commande sur le bon à partir duquel l'utilisateur enregistrera la commande;
Opérations de manipulation :

- l'utilisateur peut sélectionner le message, entrer une valeur pour num_produit, la corriger, l'effacer; il peut faire de même pour la quantité.

Client_non_valide

Nom : client_non_valide;
Définition : le message client_non_valide est le texte qui apparaît lorsque l'opérateur a introduit un numéro de client inexistant;
Type : message d'affichage;
Justification : le message a pour but de faire connaitre à l'utilisateur que le numéro de client introduit n'existe pas; il fait part d'une erreur commise;
Structure de données :

- un texte expliquant que le numéro du client introduit n'existe pas : "*Le numéro de client que vous avez introduit n'existe pas. Veuillez essayer avec une nouvelle valeur*";

Contraintes sur la présentation : la présentation et la position du message devront être similaires aux autres messages d'erreur;
Opérations de manipulation :

- l'utilisateur ne peut que supprimer ou ranger ce message;

Provenance_non_valide

Nom : provenance_non_valide;
Définition : le message provenance_non_valide signale à l'opérateur que le client dont il a introduit les références est de catégorie "douteux" pour l'entreprise;
Type : message d'affichage;
Justification : le message a pour but de faire connaitre à l'utilisateur que le client est "douteux"; il lui fait part de l'erreur commise;
Structure de données :

- un texte signalant que client introduit est "douteux" : "*Le numéro que vous avez introduit correspond à un client faisant partie de la liste noire. Il ne peut plus effectuer de commande*";

Contraintes sur la présentation : la présentation et la position du message devront être similaires aux autres messages d'erreur;

Opérations de manipulation :

- l'utilisateur ne peut que supprimer ou ranger ce message;

Produit_non_valide

Nom : produit_non_valide;

Définition : le message produit_non_valide signale à l'opérateur qu'il a introduit un numéro de produit inexistant;

Type : message d'affichage;

Justification : le message a pour but de faire connaitre à l'utilisateur que le numéro de produit introduit n'existe pas; il lui fait part de l'erreur commise;

Structure de données :

- un texte signalant que le numéro de produit introduit n'existe pas : "*Le numéro que vous avez introduit ne correspond à aucun produit existant. Veuillez essayer une nouvelle valeur*";

Contrainte sur la présentation : la présentation et la position du message devront être similaires aux autres messages d'erreur;

Opérations de manipulation :

- l'utilisateur ne peut que supprimer ou ranger ce message;

Ligne_non_valide

Nom : ligne_non_valide;

Définition : le message ligne_non_valide signale à l'opérateur que la quantité du produit commandé dépasse la quantité disponible;

Type : message d'affichage;

Justification : le message a pour but de faire connaitre à l'utilisateur que le produit de la ligne est épuisé; il lui fait part de l'erreur commise;

Structure de données :

- un texte signalant que le produit est épuisé : "*Le produit que vous avez introduit ne peut être commandé dans les quantités spécifiées*";

Contrainte sur la présentation : la présentation et la position du message devront être similaires aux autres messages d'erreur;

Opérations de manipulation :

- l'utilisateur ne peut que supprimer ou ranger ce message;

Commande_non_valide

Nom : commande_non_valide;

Définition : le message commande_non_valide signale à l'opérateur que le client ne peut passer une commande d'un tel montant;

Type : message d'affichage;

Justification : le message a pour but de faire connaitre à l'utilisateur que le montant total de commande, réduit du rabais éventuel, est supérieur à la limite d'achat du client; il lui fait part de l'erreur commise;

Structure de données :

- un texte signalant que le montant total de la commande est supérieur à la limite d'achat du client : "*La commande est refusée car son montant dépasse la limite d'achat du client.*" ;

Contrainte sur la présentation : la présentation et la position du message devront être similaires aux autres messages d'erreur;

Opérations de manipulation :

- l'utilisateur ne peut que supprimer ou ranger ce message;

Commande_enregistrée

Nom : commande_enregistrée;

Définition : le message commande_enregistrée correspond aux informations qui ont été enregistrées suite à la validation de la commande;

Type : message d'affichage;

Justification : le message a pour but de montrer à l'utilisateur la commande validée; il lui affiche l'ensemble des informations qui la constitue;

Structure de données :

- un numéro_de_commande, une date, un montant total et rabais;
- un num_client et adr_domicile;
- des lignes : qu_commandée, montant et produit;

Contrainte sur la présentation : le message devra être semblable au bon de commande manipulé par l'utilisateur;

Opérations de manipulation :

- l'utilisateur peut sélectionner ou ranger ce message;

VI.1.1.2. Remarques à propos de la spécification des messages interactifs fonctionnels

La liberté qui est offerte à l'utilisateur d'entrer les coordonnées d'un nouveau client (par son nom et son adresse) ou d'un ancien (par son numéro) implique la spécification de deux messages (Num_client et Nom_client) dont un seul sera saisi lors du déroulement de l'application interactive. C'est une structure conditionnelle déterminée par l'utilisateur[1].

Les contraintes de présentation portant sur les messages de saisie peuvent être regroupées en une seule : la présentation de ces messages à l'utilisateur devra ressembler au bon de commande qu'il enregistre.

Les messages interactifs fonctionnels que nous avons définis peuvent être décomposés en trois groupes : les messages de saisie que l'utilisateur devra fournir, les messages d'erreur et le message terminal (Commande_enregistrée) s'affichant lorsque les informations nécessaires aux fonctions ont été introduites avec succès.

La justification d'un message interactif est l'argumentation du regroupement des informations qui le constituent. Ce regroupement est fortement influencé par les messages fonctionnels à partir desquels ils sont définis. En effet, la phase de spécification fonctionnelle précède la définition de l'interface. Le concepteur de cette dernière sera, volontairement ou non, influencé par les résultats de la première phase. La plupart de nos messages interactifs

[1] Cfr. IV.1.4.2.

La justification d'un message interactif est l'argumentation du regroupement des informations qui le constituent. Ce regroupement est fortement influencé par les messages fonctionnels à partir desquels ils sont définis. En effet, la phase de spécification fonctionnelle précède la définition de l'interface. Le concepteur de cette dernière sera, volontairement ou non, influencé par les résultats de la première phase. La plupart de nos messages interactifs fonctionnels sont équivalents aux messages fonctionnels[1]. Leur justification peut être exprimée globalement par le fait que le concepteur considère le regroupement fonctionnel justifié d'un point de vue ergonomique ou psychologique.

L'influence de la découpe des messages fonctionnels sur la définition des messages interactifs fonctionnels ne doit pas entraîner l'inadaptation de l'interface à l'utilisateur. L'attention du concepteur devra avant tout être centrée sur ce dernier.

VI.1.1.3. Schéma de la conversation

Le schéma de la conversation spécifie l'ordre de traitement des messages interactifs fonctionnels. Nous voyons à la figure VI.1. que l'opérateur a la liberté de saisir dans l'ordre qu'il désire les messages Num_client ou Nom_client et un nombre quelconque de Ligne_cmde (ce nombre sera déterminé par l'opérateur en fonction du bon de commande qu'il enregistre).

S'il introduit le message Num_client, il verra apparaître le message Client_invalide si le numéro qu'il a saisi n'existe pas ou le message Provenance_invalide si le client auquel correspond le numéro est de catégorie "douteux". Dans le cas contraire, il pourra saisir Adr_domicile si le client a changé d'adresse.

S'il introduit le message Nom_client, il devra ensuite saisir le message Adr_domicile.

Il pourra introduire un nombre quelconque de messages Ligne_cmde; chaque message participera au point de synchronisation si le numéro de produit saisi existe et que la quantité commandée est disponible ou qu'il existe un produit de substitution en quantité suffisante.

La réalisation du point de synchronisation peut donc s'exprimer de la manière suivante :

- la saisie d'un message Num_client dont le numéro et la provenance sont valides,
ou
- la saisie de Adr_domicile,
et
- le signalement par l'opérateur de la fin de la saisie des messages Ligne_cmde.

[1] Ils le sont touts à l'exception du message ligne_de_commande regroupant les messages num_produit et qu_produit.

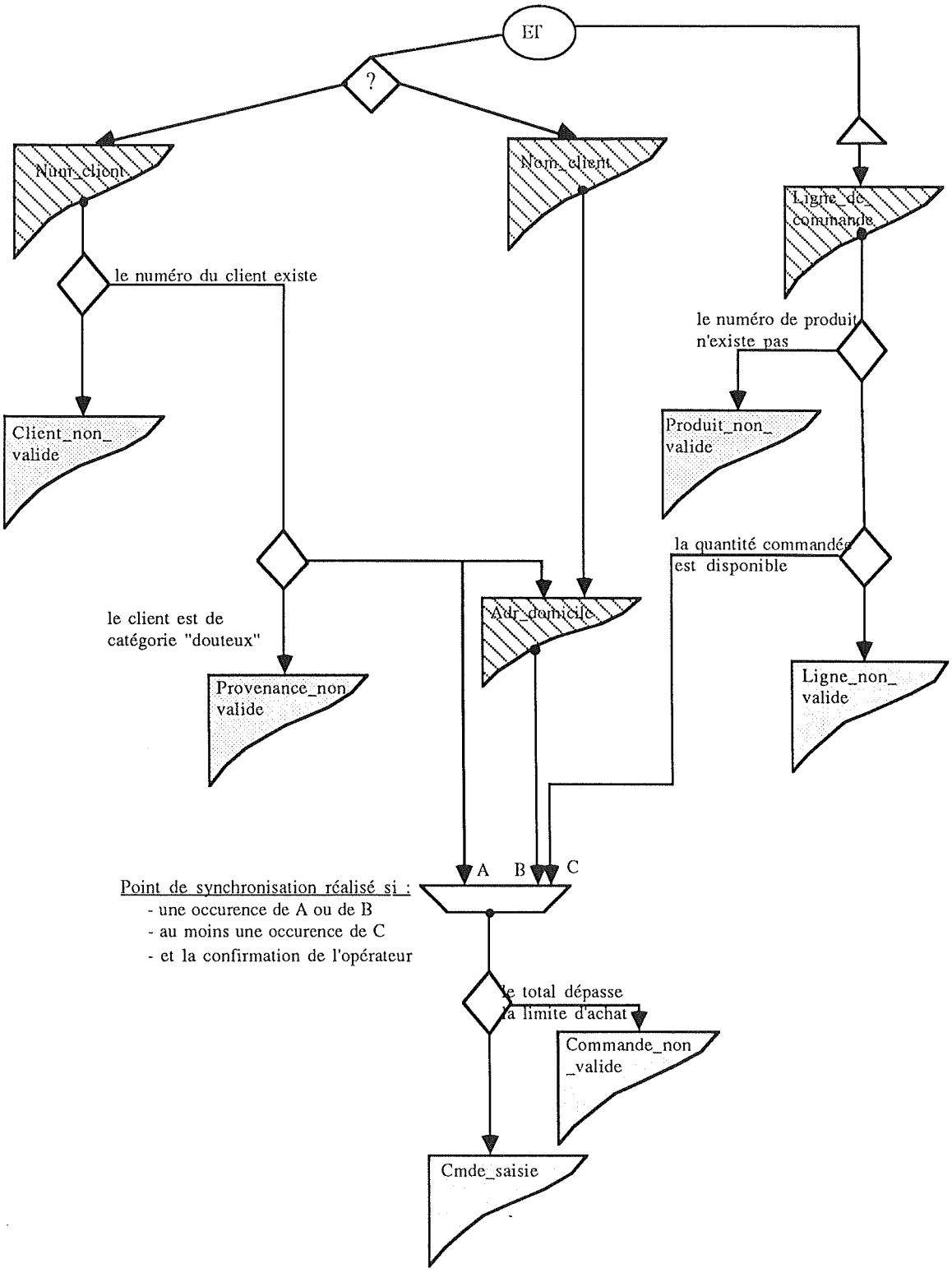


Figure VI.1. : schéma de la conversation de la première interface

Nous avons un point de synchronisation demandant une intervention explicite de l'opérateur. Ceci est dû au nombre indéterminé de messages Ligne_cmde qui peuvent être introduits. Un mécanisme devra être offert pour permettre le signalement de la terminaison de la saisie des lignes de commande.

La réalisation du point de synchronisation entraînera l'affichage du message Commande_invalide si le total de la commande dépasse la limite d'achat du client. Sinon, le message Cmde_saisie apparaîtra et la commande aura été enregistrée.

Notons que le schéma est incomplet car l'affichage de messages d'erreur provoquera un retour dans le schéma à l'endroit qui précede le message de saisie ayant entraîné l'erreur. Dans un soucis de clarté, nous avons préféré ne pas les représenter et les considérer comme implicites.

VI.1.1.4. Spécification des messages purement interactifs

Un message purement interactif ne concerne que l'interface; les fonctions n'en n'ont pas connaissance. Voici les messages que nous avons spécifiés pour la saisie d'une commande écrite :

a) Messages de contrôle

- Saisie terminée :

Nom : saisie_terminée;
Définition : ce message permet à l'utilisateur de signaler à l'interface qu'il a terminé l'introduction des lignes de la commande;
Type : message de contrôle;
Est déclenché : il peut l'être lorsque l'opérateur a introduit une ligne de commande valide;
Justification : ce message est nécessaire pour pouvoir satisfaire le point de synchronisation avant de valider la commande; c'est en effet l'opérateur qui signale la fin de la saisie des lignes de commande (il détermine leur nombre)
Contenu : un libellé explicatif;
Contraintes sur la présentation : le message devra être présenté par une icône ou une commande, manipulable dès qu'une ligne de commande a été validée;
Opérations de manipulation :
- clôturer le message;

- Saisie annulée :

Nom : saisie_annulée;
Définition : ce message permet à l'opérateur de signaler au dialogue l'annulation de la commande qu'il était en train d'introduire;
Type : message de contrôle;
Est déclenché : à n'importe quel moment lorsque l'utilisateur le désire;
Justification : Ce message donne la possibilité à l'utilisateur d'annuler des saisies erronées et qui ne peuvent être modifiées autrement; il constitue une unité de saisie pour l'opérateur car il lui permet de signaler à l'interface son désir d'annuler la saisie de la commande en cours;
Contenu : un texte explicatif;

Contraintes sur la présentation : le message devra être présenté par une icône visible en permanence ou une commande possible à tous moment;
Opérations de manipulation :
- sélectionner et clôturer le message

- Quitter_saisie :

Nom : quitter_saisie;
Définition : ce message permet à l'opérateur de signaler au dialogue qu'il annule la commande qu'il était en train d'introduire et qu'il désire quitter l'application interactive;
Type : message de contrôle;
Est déclenché : à n'importe quel moment lorsque l'utilisateur le désire;
Justification : ce message donne la possibilité à l'utilisateur de quitter l'application interactive quelque soit sa position dans le schéma de la conversation;
Contenu : un texte explicatif;
Contraintes sur la présentation : le message devra être présenté par une icône visible en permanence ou une commande activable à tout moment;
Opérations de manipulation :
- clôturer le message

- Choix_saisie

Nom : choix_saisie;
Définition : ce message permet à l'opérateur de choisir d'introduire soit le Nom_client, soit le Num_client, soit les Lignes_cmde;
Type : message de contrôle;
Est déclenché : au début de la conversation et lorsqu'une saisie contribution au point de synchronisation;
Justification : ce message permettra à l'utilisateur de choisir le message qu'il désire saisir. Il pourra saisir le Nom_client ou le Num_client (un des deux) ou les Lignes_de_commande. Ce message devra apparaître chaque fois que l'opérateur aura effectué l'une de ces saisies (qui contribuera au point de synchronisation). Ce message sera inutile dans un contexte de manipulation directe offrant à l'utilisateur la possibilité de sélectionner le message qu'il désire saisir;
Contenu : les alternatives offertes (choix de la saisie du Nom_client, Num_client ou des Lignes_cmde);
Contraintes sur la présentation : il serait souhaitable d'offrir la possibilité à l'opérateur d'effectuer son choix sans l'aide de ce message. Si l'environnement ne le permet pas, le message pourra apparaître sous forme de menu;
Opérations de manipulation :
- sélectionner l'alternative choisie et clôturer le message;

b) Messages d'erreur syntaxique

Voici les contraintes syntaxiques que devra vérifier l'interface future (qu'elle interdise la faute ou qu'elle envoie un message d'erreur).

- Num_client_entier :

Nom : num_client_entier;
Définition : ce message a pour but de faire connaître à l'opérateur qu'il a introduit un numéro de client non entier;
Type : message d'erreur syntaxique;
Est déclenché : après la saisie du message Num_client si le numéro du client n'est pas un entier;
Justification : ce message constitue une unité d'affichage pour l'utilisateur puisqu'il l'informe d'une erreur commise;
Contenu : un texte : "*le numéro de client que vous avez entré doit être un entier*";
Contraintes sur la présentation : ce message pourra être remplacé par une gestion des caractères introduits pour Num_client de sorte que seuls les chiffres soient acceptés. De plus, il pourrait être supprimé en considérant qu'il s'agit alors d'un numéro de client inexistant (affichage du message Client_invalide);
Opérations de manipulation : supprimer ou ranger le message;

- Adr_dom_NPA_entier :

Nom : adr_dom_NPA_entier;
Définition : ce message a pour but de faire connaître à l'opérateur qu'il a introduit un NPA non entier;
Type : message d'erreur syntaxique;
Est déclenché : après la saisie du message Adr_domicile si le NPA de l'adresse n'est pas un entier;
Justification : ce message constitue une unité d'affichage pour l'utilisateur puisqu'il l'informe d'une erreur commise;
Contenu : un texte signalant que le NPA introduit doit être un entier " *le code postal de l'adresse doit être un nombre. Veuillez introduire une nouvelle valeur.*";
Contraintes sur la présentation : le message devra expliquer clairement l'erreur commise;
Opérations de manipulation : supprimer ou ranger le message;

- Saisie_référence_client :

Nom : saisie_référence_client;
Définition : ce message a pour but de faire connaître à l'utilisateur qu'il doit introduire les références du client qui passe la commande;
Type : message d'erreur syntaxique;
Est déclenché : lorsque l'utilisateur veut valider la commande (il clôture le message Saisie_terminée) sans avoir saisi le message Num_client ou le message Nom_client;
Justification : cette erreur est susceptible de se produire dans un contexte de manipulation directe où l'opérateur sélectionne le message qu'il souhaite saisir. Il constitue une unité d'affichage pour l'utilisateur car il l'informe d'une erreur commise. Il est inutile si l'opérateur est contraint de saisir le message Choix_saisie car il devra alors saisir l'une des deux alternatives;

Contenu : un texte signalant à l'opérateur qu'il doit saisir soit le numéro du client, soit son nom et son adresse s'il s'agit d'un nouveau client : "*Veuillez introduire le nom ou le numéro du client*";
Contraintes sur la présentation : - non définies -
Opérations de manipulation : supprimer ou ranger le message;

- Quantité_entière_positive :

Nom : quantité_entière_positive;
Définition : ce message a pour but de faire connaître à l'opérateur qu'il a introduit une quantité à commander non entière;
Type : message d'erreur syntaxique;
Est déclenché : après la saisie du message Ligne_de_commande si la quantité à commander du produit n'est pas un entier;
Justification : ce message constitue une unité d'affichage pour l'utilisateur puisqu'il l'informe de l' erreur commise;
Contenu : un texte signalant que la quantité à commander doit être un entier : "*Veuillez introduire une quantité entière positive*";
Contraintes sur la présentation : ce message pourra être remplacé par une gestion des caractères introduits pour la quantité à commander de sorte que seuls les chiffres soient acceptés. De plus, il peut être supprimé en considérant qu'il s'agit d'une erreur entraînant l'affichage du message Ligne_invalide;
Opérations de manipulation : supprimer ou ranger le message;

- Num_produit_incorrect :

Nom : num_produit_incorrect;
Définition : ce message a pour but de faire connaître à l'opérateur qu'il a introduit un numéro de produit incorrect car il n'est pas entier ou la division des 7 premiers par 7 n'est pas égale au huitième;
Type : message d'erreur syntaxique;
Est déclenché : après la saisie du message Ligne_commande si le numéro de produit n'est pas un entier ou que le dernier chiffre n'est pas égal au reste des 7 premiers chiffres par 7;
Justification : ce message constitue une unité d'affichage pour l'utilisateur car il l'informe de l'erreur commise;
Contenu : un texte signalant que le numéro du produit introduit est incorrect : "*Le numéro de produit que vous avec est incorrect. Veuillez taper un nouveau numéro*";
Contraintes sur la présentation : le message pourrait être supprimé en considérant qu'il s'agit d'un numéro de produit inexistant et en affichant le message Produit_invalide;
Opérations de manipulation : supprimer ou ranger le message;

c) Messages d'aide

Nous avons décidé de nous contenter d'offrir à l'utilisateur un minimum d'aide car notre but n'est pas de créer une application interactive de haut niveau mais bien de tester notre méthodologie sur un exemple relativement complexe. La spécification de messages d'aide incluant la gestion du contexte, d'un tableau de bord... se fera de la même manière que le message d'aide que nous spécifions ci-dessous. Ils contiendrons néanmoins des informations manipulées par les fonctions. Nous analyserons leur incidence à la section VI.1.3.

- aide_sémantique :

Nom : aide_sémantique;
Définition : ce message affiche à l'opérateur des informations suceptibles de mieux lui faire comprendre la sémantique de l'application interactive;
Type : message d'aide;
Est déclenché : à n'importe quel moment, lorsque l'opérateur le désire. Pour cela, il devra sélectionner le message;
Justification : ce message constitue une unité d'affichage car il présente à l'utilisateur un ensemble d'informations concernant les fonctionnalités de l'application interactive;
Contenu : un texte expliquant les mécanismes qui régissent l'application proprement dite;
Contraintes sur la présentation : le message pourra être décomposé en plusieurs partie, offrant à l'utilisateur le choix de poursuivre la lecture ou de revenir à l'exécution de l'application.
Opérations de manipulation :
- sélectionner et supprimer ou ranger le message;

VI.1.1.5. Remarques sur la spécification des messages interactifs et le schéma de la conversation

a) Messages sans structure de données

Une première remarque concerne les messages de saisie ne contenant pas d'information à introduire par l'opérateur comme les messages Saisie_annulée, Saisie_terminée ou Quitter_saisie. Le message ne contiendra pas d'information manipulable mais pourra être clôturé par l'utilisateur lorsqu'il souhaite le communiquer à l'interface[1]. Ce type de message est la représentation d'une **opération** qui ne porte pas sur un message particulier mais sur un ensemble de messages (Saisie_terminée) et/ou le schéma de la conversation (Saisie_annulée, Quitter_saisie et Saisie_terminée).

b) Messages de contrôle communs à toute application interactive

Dans le but d'offrir la possibilité de quitter l'application interactive ou d'annuler les traitements en cours, il nous a fallu ajouter deux messages interactifs de contrôle qui ne peuvent pas être déduits du schéma de la conversation. Ils sont en réalité indépendants des caractéristiques du schéma : le premier (Quitter_saisie) permet de le quitter et par conséquent d'abandonner le déroulement de l'application. Le second (Annuler_saisie) permet de se replacer à son début. Ils doivent exister pour toute application interactive et dès lors être disponibles

[1] Ces messages possèdent une sémantique implicite, connue de l'interface.

quelque soit l'environnement utilisé. Nous avons décidé de les spécifier pour souligner leur existence.

c) Parallélisme

Le schéma de la conversation permet de situer l'opérateur à un endroit précis du déroulement de l'application interactive. La structure parallèle introduit une liberté supplémentaire : l'utilisateur choisit l'ordre dans lequel il effectue ses traitements. Pour notre exemple, il peut saisir le nom ou le numéro du client et les lignes de commande dans l'ordre qu'il souhaite. Dans un environnement présentant au même moment l'ensemble des données à introduire et permettant de voyager entre celles-ci (comme par exemple la manipulation directe), l'opérateur aura la possibilité de se situer à plusieurs endroits en même temps dans la structure parallèle. Il pourra par exemple introduire le nom du client, puis entrer une ligne de commande, ensuite passer au prénom et au titre du client... Il ne sera alors plus possible de situer l'utilisateur à un seul endroit mais à autant d'endroits que la structure parallèle possède de degré de liberté. A la figure VI.2. l'opérateur pourra se situer à 3 endroits différents au même moment[1]. Il devra bien sûr respecter l'enchaînement décrit à l'intérieur de chaque chemin quittant la structure parallèle[2].

Cette remarque nous semblait importante pour une bonne lecture du schéma. Remarquons cependant que tous les environnements ne permettent pas cette possibilité. Le message interactif de contrôle choix_saisie sera alors utilisé pour sélectionner le message que l'on désire traiter et le degré de liberté sera constamment de 1[3].

En conséquence, la lecture du schéma de la conversation sera dépendante du style d'interaction qui sera utilisé pour l'implémenter.

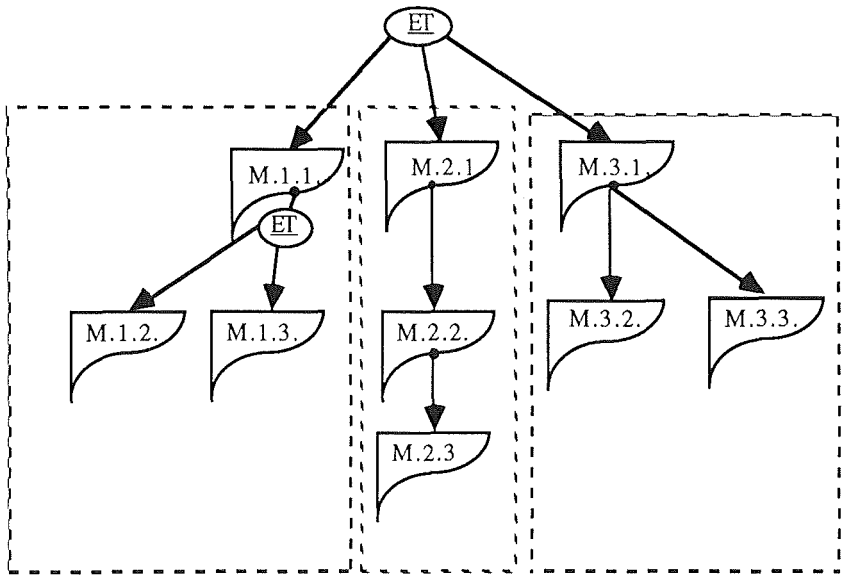


Figure 6.2. Parallélisme : l'utilisateur peut se situer à 3 endroits différents dans le schéma de la conversation (un endroit par sous-graphe).

[1] En réalité, il pourra même se situer à quatre endroits différents car nous avons une structure parallèle emboîtée : la fin du traitement du message M 1,1 déclenche la saisie en parallèle des messages M 1,2 et M 1,3.

[2] La structure parallèle entraîne la création de sous-graphes : chaque flèche quittant la structure est le, déclenchement d'un sous-graphe dans lequel l'utilisateur évoluera indépendamment du reste du schéma.

[3] L'utilisateur devra progresser jusqu'à la fin du sous-graphe pour pouvoir se déplacer dans un autre sous-graphe.

d) La synchronisation

Nous avons spécifié un message d'erreur syntaxique lié à la conversation (Saisie_référence_client). Son existence dépend de la nature du point de synchronisation : nous avons signalé[1] qu'il était nécessaire que l'utilisateur avertisse l'interface de la fin de la saisie des lignes de commande car leur nombre lui était inconnu. La synchronisation pourra également se faire de manière explicite pour le point de synchronisation global (la saisie des références du client[2] et les lignes de commande). L'opérateur signalera à l'interface qu'il considère le point de synchronisation comme réalisé. Il faudra alors effectuer la vérification de cette information et afficher si nécessaire un message d'erreur. Ce type de synchronisation a pour avantage de permettre la correction d'un message participant à la synchronisation après l'avoir clôturé. Par exemple, l'utilisateur pourra changer l'adresse du client tant qu'il n'a pas expressément signaler la réalisation de la synchronisation.

En résumé, le message Saisie_référence_client existera si la synchronisation globale est signalée par l'opérateur[3].

VI.1.2. Spécification pour une commande téléphonée

VI.1.2.1. Spécification des messages interactifs fonctionnels

Nous ne spécifierons ici que les messages interactifs différents de notre première interface. Leur existence est liée aux caractéristiques différentes d'une commande téléphonée par rapport à un bon de commande.

Au contraire d'une commande écrite, l'enregistrement d'une commande téléphonée met également en interaction le client qui passe la commande. Ce dernier est l'utilisateur final de l'application interactive, le destinataire des informations diffusées par l'interface. Dès lors, il est indispensable d'améliorer la qualité des informations qui seront communiquées pour permettre au client de passer une commande valide[4].

Les éléments supplémentaires que l'interface devra communiquer sont les suivants :

- l'affichage du **nom et de l'adresse du client** après l'introduction du numéro : il permet la validation du numéro introduit ("*Vous appelez-vous bien Pierre Durant?*") et la vérification de l'adresse (le client peut avoir déménagé sans en avoir averti la firme). Ces informations favoriseront la satisfaction du client qui se sentira en de bonnes mains de voir que l'entreprise dispose de moyens informatiques élaborés.

- l'affichage du **numéro du client** après l'introduction de son nom, prénom et titre : ceci est très important car il permet de communiquer au client le numéro par lequel il devra dorénavant passer des commandes.

---

[1] Cfr section VI.1.1.3.

[2] Son numéro ou son nom et adresse.

[3] Voir section VI.4.1.2.d.

[4] Ces informations sont communiquées à l'opérateur mais en sortie de la phase. Ceci n'est pas suffisant puisque la commande est alors enregistrée et donc clôturée.

- l'affichage du **total** et éventuellement du **rabais** de la commande : le client sera naturellement désireux de connaître la somme qu'il devra payer à la réception de sa facture et également du rabais qui lui est offert s'il est membre du personnel;

- l'affichage du **libellé** de chaque produit commandé : l'opérateur qui saisissait une commande à partir d'un bon de commande pouvait considérer une ligne de commande comme une unité de saisie. Par contre, pour la saisie d'une commande téléphonée, il est nécessaire d'avoir un "feedback" immédiat (le libellé) après la saisie du numéro de produit et avant de spécifier la quantité (l'opérateur peut alors dialoguer avec le client pour savoir si le produit est bien celui qu'il désire). On éclatera donc le message interactif fonctionnel "ligne_de_commande" en deux messages interactifs : "num_produit" et "qu_produit". La satisfaction du client sera également améliorée car il sera sûr que les produits qu'il commande sont ceux qu'il souhaitait.

Une autre différence se fera dans la qualité des informations du message Commande_invalide. Ce message contiendra à présent la limite d'achat du client qui a été dépassée. Cette information n'intéressait pas l'opérateur dans la première interface car il n'était pas en communication avec le client. Ce dernier sera par contre intéressé de la connaître pour pouvoir par exemple supprimer une ligne de commande et ne plus la dépasser.

En résumé, les informations supplémentaires que nous allons spécifier sont avant tout destinées au client qui passera la commande plutôt qu'à l'opérateur auquel elles seront communiquées. Les messages interactifs fonctionnels différents de la première interface sont donc :

Num_client_à_afficher

Nom : num_client_à_afficher;
Définition : un numéro_client est un numéro attribué par compostage lors de la création du client. Il est identifiant;
Type : message d'affichage;
-- ce message sera affiché lorsque l'opérateur aura introduit le nom d'un nouveau client;
Justification : Ce message constitue une unité d'affichage pour l'utilisateur car il le renseigne du numéro que le nouveau client devra dorénavant fournir pour passer de nouvelles commandes;
Structure de données :
num_client : entier;
Contraintes sur la présentation : la présentation pourrait être semblable au bon de commande si l'opérateur futur employera les deux interfaces (et ainsi lui éviter des incohérences d'uniformité);
Opérations de manipulation :
- aucune opération de manipulation car seule l'interface gère ce message[1];

Nom_client_à_afficher

Nom : nom_client_à_afficher;
Définition : un nom_client est l'identité du client connue par la firme;
Type : message d'affichage;

[1] L'utilisateur verra l'affichage du message après la saisie d'un numéro de produit valide. Il constitue une information de validation supplémentaire du numéro de produit.

-- ce message sera affiché si l'utilisateur a saisi le message num_client_à_saisir (il s'agit alors d'un ancien client);
Justification : ce message constitue une unité d'affichage pour l'opérateur car il constitue les informations nécessaires à l'utilisateur pour vérifier que le numéro de client introduit correspond effectivement au client passant la commande (le numéro n'est pas erroné);
Structure de données :

. nom : char[40];
. prénom : char[40];
. titre : char[40];

Contraintes sur la présentation : la présentation pourrait être semblable au bon de commande si l'opérateur futur employera les deux interfaces (et ainsi lui éviter des incohérences d'uniformité);
Opérations de manipulation :

- aucune opération de manipulation car seule l'interface gère ce message;

Adr_domicile_à_afficher

Nom : adr_domicile_à_afficher;
Définition : une adresse d'un client est constituée d'une rue, un numéro, un NPA et d'une localité;
Type : message d'affichage;
-- ce message apparaîtra après la saisie du numéro de client pour indiquer à l'utilisateur la dernière adresse connue du client;
Justification : ce message constitue une unité d'affichage pour l'opérateur car il représente l'endroit où devront être acheminés les commandes ou l'endroit où réside le client;
Structure de données :

. rue : char[40];
. numéro : char[6];
. NPA : char[6];
. localité : char[40];

Contraintes sur la présentation : la présentation pourrait être semblable au bon de commande si l'utilisateur futur employera les deux interfaces (et ainsi lui éviter des incohérences d'uniformité);
Opérations fonctionnelles :

- aucune opération fonctionnelle car l'application est seule à gérer ce message;

Num_produit

Nom : Num_produit;
Définition : le message est constitué d'un numéro à l'aide duquel l'entreprise répertorie ses produits;
Type : message de saisie;
Justification : ce message a pour but de saisir le numéro du produit que le client désire commander; il constitue une unité de saisie pour l'opérateur car celui-ci doit être capable de vérifier que le numéro de produit correspond bien à la volonté du client;
Structure de données : num_produit : entier;
Contrainte d'intégrité :

- le numéro de produit doit être un entier positif et la division des premiers chiffres par 7 doit être égale au huitième ;

Contraintes sur la présentation : dans le but de favoriser la tâche de l'utilisateur, il y aura un feedback immédiat du libellé du produit correspondant si le numéro de produit est valide;
Opérations de manipulation :

- l'opérateur peut sélectionner le message, affecter une valeur à son attribut, la corriger ou l'effacer.

Qu_produit

Nom : qu_produit;
Définition : le message représente la quantité de produit que le client passant la commande désire;
Type : message de saisie;
Justification : ce message a pour but de saisir la quantité du produit à commander si le client la spécifie; il constitue une unité de saisie car il permet à l'utilisateur de spécifier la quantité du produit précédemment entré et dont il connait à présent le libellé;
Structure de données : qu_produit : entier strictement positif;
Contraintes d'intégrité :

- la quantité doit être un entier positif;

Contraintes sur la présentation : la présentation pourrait être semblable au bon de commande si l'opérateur futur employera les deux interfaces (et ainsi lui éviter des incohérences d'uniformité);
Opérations de manipulation :

- l'utilisateur peut affecter une valeur à l'attribut du message, la corriger ou l'effacer.

Libellé_produit

Nom : libellé_produit;
Définition : le libellé_produit représente la définition d'un produit sour forme de texte;
Type : message d'affichage;
Justification : ce message a pour but de produire à l'opérateur un feedback pour que celui-ci connaisse le libellé du produit dont il a tapé le numéro; il constitue donc une unité d'affichage puisqu'il représente une vérification textuelle du numéro de produit introduit;
Structure de données : libellé_produit : char[50];
Contraintes sur la présentation : libellé_produit devra être affiché bien en correspondance avec le numéro de produit num_produit pour éviter à l'utilisateur un effort de visualisation (le libellé devra s'afficher visuellement près de num_produit);
Opérations de manipulation :

- l'opérateur ne peut pas manipuler ce message car il est entièrement géré par l'interface;

Commande non valide

Nom : commande_non_valide;
Définition : une commande_non_valide est le message d'erreur qui apparaîtra lorsque le montant de la commande dépasse la limite d'achat du client;
Type : message d'affichage;
Justification : le message a pour but de faire connaitre à l'utilisateur que le montant total de commande, réduit du rabais éventuel, est supérieur à la limite d'achat du client; il constitue une unité d'affichage pour l'opérateur puiqu'il lui fait part de l'erreur commise;
Structure de données :

- un texte signalant que le montant total de la commande est supérieur à la limite d'achat du client en lui communiquant cette limite;

Contrainte sur la présentation : le texte du message devra expliquer clairement l'erreur commise en affichant la limite d'achat;

Opérations de manipulation :

- l'utilisateur ne peut que supprimer ou ranger ce message;

Total

Nom : total;
Définition : le message représente le montant de la commande, en déduisant le rabais éventuel;
Type : message d'affichage;
Justification : le message a pour but de faire connaître à l'opérateur le montant total de la facture qu'il a saisie; il constitue une unité d'affichage pour l'utilisateur puisqu'il correspond au montant que le client devra effectivement débourser pour la commande passée;
Structure de données :

- un réel : montant_total;

Contraintes sur la présentation : - non définies -
Opérations de manipulation:

- l'opérateur ne peut pas manipuler ce message car il est entièrement géré par l'interface;

Rabais

Nom : rabais;
Définition : le message représente le montant de rabais offert au client;
Type : message d'affichage;
Justification : le message a pour but de faire connaître à l'utilisateur le montant du rabais qui lui est offert s'il est membre du personnel; il constitue une unité d'affichage pour l'opérateur car ce montant n'est accordé qu'aux membres du personnel, ce qui constitue une unité d'information;
Structure de données : un réel : montant_rabais;
Contraintes sur la présentation : - non définies -
Opérations de manipulation :

- l'utilisateur ne peut pas manipuler ce message car l'interface est seule à le gérer;

VI.1.2.2. Remarques sur les spécifications des messages interactifs fonctionnels

a) Messages supplémentaires

Nous avons ajouté un nombre assez important de nouveaux messages pour offrir au client toutes les informations qui pouvaient affecter son comportement et lui permettre de terminer la communication téléphonique en ayant passé une commande valide répondant au mieux à ses désirs.

b) Messages interactifs et messages fonctionnels

Si nous anticipons quelque peu sur la phase de validation de la tâche par rapport aux traitements, nous constatons que les nouvelles informations que nous venons de spécifier ne font pas partie des messages fonctionnels de l'application proprement dite. Celle-ci ne fournit en effet aucun élément informatif sur le déroulement de ses traitements avant l'enregistrement de la commande c'est-à-dire en sortie de la phase. Or, nous avons expliqué l'importance de la disponibilité de ces renseignements. Il est alors nécessaire de modifier la spécification des messages fonctionnels provenant de l'application pour réaliser notre objectif. Ces informations ne sont pas accessibles à l'interface car elles font partie de la base de données. Nous avons montré à la section III.1. que seule l'application proprement dite y avait accès. Ce manque d'informations peut être considéré comme une erreur de spécification. Nous traitons de ce problème à la section VI.1.3.

c) Découpe en messages interactifs

La découpe des messages interactifs fonctionnels de notre deuxième interface correspond exactement aux messages fonctionnels de l'application proprement dite (du moins en ce qui concerne les informations qui s'y retrouvent[1]). Les fonctions de l'application s'exécuteront plus rapidement si la saisie d'un message interactif fonctionnel provoque le déclenchement d'une fonction. Examinons pour cela le travail du module de la correspondance : il n'enverra un message fonctionnel à l'application proprement dite que si toutes les informations du message ont été saisies. Les fonctions s'exécuteront donc plus rapidement si la saisie d'un seul message interactif fonctionnel provoque l'envoi du message fonctionnel correspondant.

L'objectif d'efficacité de la deuxième interface est ainsi rencontré.

d) Contraintes sur la présentation

Nous avons suggéré dans les contraintes sur la présentation des messages d'utiliser la même visualisation pour les deux interfaces. Elle permettra de leur uniformité. Celle-ci est souhaitable si l'opérateur de l'application interactive effectue les deux types de saisie. Sinon, il est préférable d'offrir à l'opérateur une présentation reflètant au mieux les caractéristiques de sa tâche[2].

VI.1.2.3. Spécification du schéma de la conversation

Le schéma de la conversation de la deuxième interface est présenté à la figure VI.3. L'opérateur devra commencer par introduire le numéro ou le nom du client. La terminaison de la saisie du nom déclenchera l'affichage du nouveau numéro du client. Si le numéro est fourni, le nom et l'adresse du client seront affichés.

Lorsque l'utilisateur a terminé de saisir les références du client, il peut alors passer aux lignes de la commande. Il en introduira le nombre qu'a décidé le client. La saisie de la quantité est facultative; elle est égale à la quantité par défaut lorsqu'elle n'a pas été précisée.

---

[1] Cfr paragraphe précédent.

[2] Un bon de commande est conçu autant pour le client que pour l'opérateur qui l'enregistrera. L'opérateur enregistrant une commande téléphonée pourra désirer une présentation tout-à-fait différente.

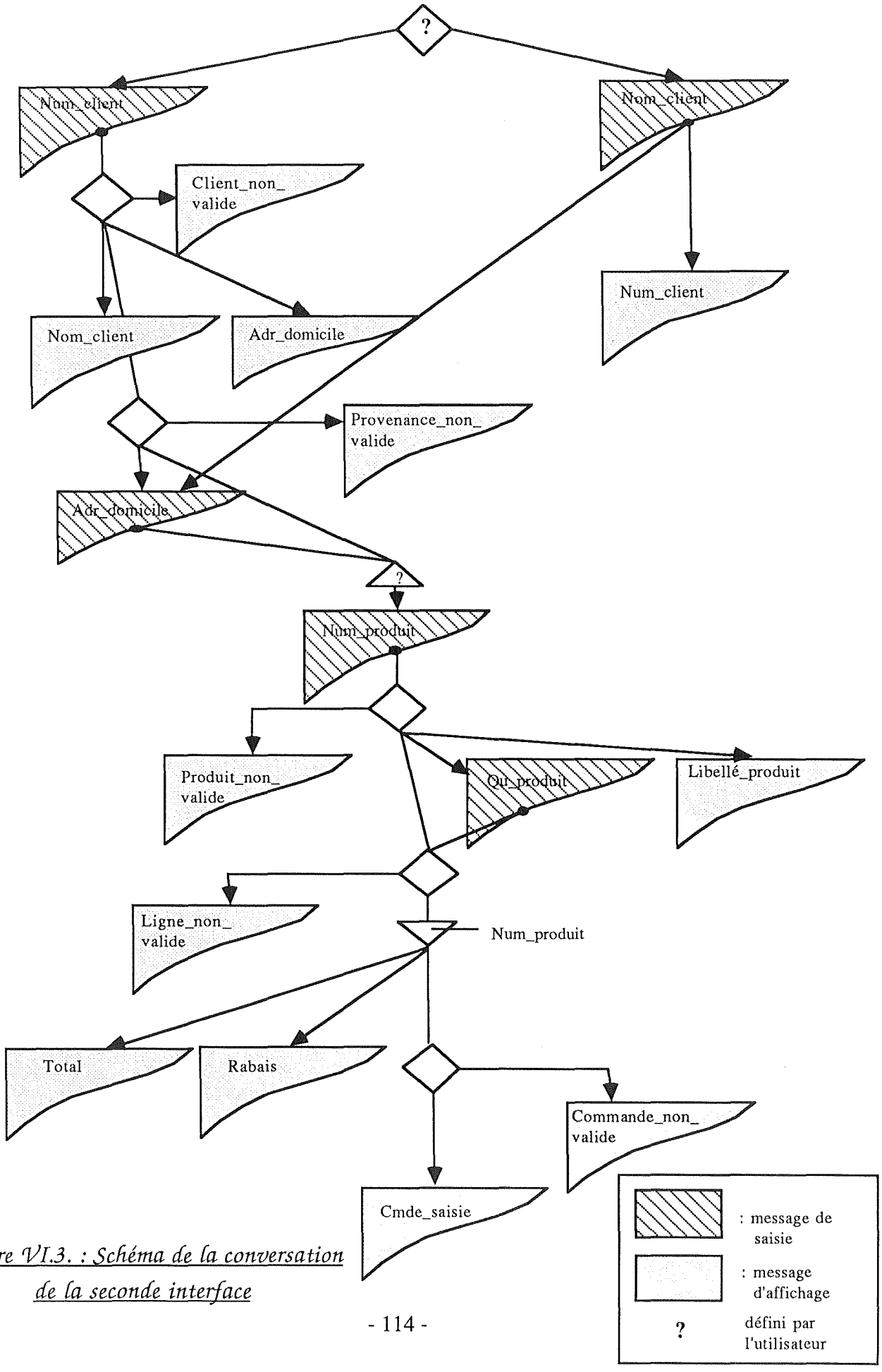


Figure VI.3. : Schéma de la conversation de la seconde interface

VI.1.2.4. Remarques sur le schéma de la conversation

Le schéma de la conversation de la seconde interface est plus restrictif que pour la première. Ceci est dû à la contrainte de temps dont nous avions parlé à la section V.1. L'opérateur travaillera de manière plus rapide si on lui évite de devoir faire des choix[1]. Il n'aura plus la possibilité de choisir de débuter par les références du client ou les lignes de commande. Ceci reste cependant logique avec la conception que se font l'opérateur et le client de la passation d'une commande : il est naturel de s'identifier avant de spécifier ce que l'on désire.

Le point de synchronisation est devenu de type accumulation : la synchronisation ne porte plus que sur les lignes de la commande. Il sera déterminé par le client

Nous avons décidé à la section VI.1.1.3. de ne pas représenter dans le schéma de la conversation les retours arrière lors d'une saisie erronée comme par exemple un numéro de client inexistant. Un autre type de retour arrière pourra survenir. Prenons l'exemple de la saisie d'un numéro de produit : la terminaison de sa saisie entraînera l'affichage du libellé correspondant. L'opérateur le communiquera au client pour lui demander confirmation. S'il ne s'agit pas du produit désiré, il est nécessaire de disposer d'un mécanisme de retour arrière pour modifier la saisie erronée. Chaque message de saisie pourra de la sorte être ultérieurement invalidé et nécessiter un retour dans le schéma de la conversation à l'endroit de sa saisie. Ce type de retour arrière peut également être considéré comme implicite car il porte sur chaque message de saisie et peut être activé à tous moment. Sa représentation dans le schéma constituerait une surcharge inutile.

La possibilité de retour arrière peut avoir une influence sur la spécification des fonctionnalités. Nous discutons de ce problème à la section VI.1.3. évaluant la spécification de la Tâche.

VI.1.2.5. Spécification des messages purement interactifs

Les messages purement interactifs ne différeront de la première interface qu'en fonction des modifications apportées au schéma de la conversation. Ceci s'explique facilement pour les messages de contrôle puisqu'ils en dépendent directement. Les erreurs syntaxiques seront les mêmes (il y a toujours obligation d'introduire une quantité entière positive, un numéro de client qui soit un nombre...) sauf s'ils se rapportent au schéma de la conversation (message Saisie_référence_client).

Par contre, les messages d'aide pourraient être tout-à-fait différents selon l'interface pour laquelle ils seront spécifiés[2]. Cependant, nous n'avons défini qu'un seul message se rapportant aux fonctions qui, elles, restent les mêmes.

---

[1] Un choix implique de sélectionner l'alternative choisie. Cette sélection entraîne une perte de temps qui peut devenir non négligeable dans des applications offrant un nombre important de possibilités à l'utilisateur.

[2] Voir section suivante.

Voici les messages qui diffèrent :

- Choix_saisie

Nom : choix_saisie;
Définition : ce message permet à l'opérateur de choisir d'introduire soit le Nom_client, soit le Num_client;
Type : message de contrôle;
Est déclenché : au début de la conversation;
Justification : ce message permettra à l'utilisateur de choisir le message qu'il désire saisir selon que le client soit nouveau ou ancien. Rappelons qu'il sera inutile dans un environnement tel que la manipulation directe;
Contenu : les alternatives offertes (choix de la saisie du Nom_client ou Num_client);
Contraintes sur la présentation : il serait souhaitable d'offrir la possibilité à l'opérateur d'effectuer son choix sans l'aide de ce message. L'utilisateur pourrait passer d'un message à l'autre à l'aide d'une fonction;
Opérations de manipulation :
- sélectionner l'alternative choisie;

- le message saisie_référence_client devient inutile. L'opérateur sera obligé de saisir les références du client.

VI.1.3. Evaluation de la spécification de la Tâche

Nous évaluons à présent nos hypothèses en fonction des remarques que nous avons pu faire tout au long de la spécification de la Tâche.

VI.1.3.1. Indépendance de conception entre les fonctionnalités et l'interface

Les retours arrière dus à des erreurs ou à la volonté de l'utilisateur ne sont pas représentés dans le schéma de la conversation car ils peuvent être considérés comme implicites. Le deuxième type de retour arrière est connu sous le nom de fonction "UNDO" par laquelle il est possible de 'défaire' certaines actions qui ont donné des résultats indésirés. Elle peut avoir un impact sur les fonctionnalités dans le cas où les opérations à défaire ont modifié des données manipulées par l'application. Par exemple, l'opérateur chargé d'enregistrer un nouveau client pourra se tromper dans sa saisie (il tape une mauvaise touche sans s'en rendre compte, il a mal compris le client...) et vouloir annuler le nouveau client enregistré. Les fonctions devront alors être capable d'effectuer cette opération. Or rien ne prouve que la phase de spécification des fonctionnalités aura mis en évidence ce besoin; elle l'aura fait après une étude détaillée des besoins de l'utilisateur. En bref, si l'indépendance modulaire entre les fonctions et l'interface n'est pas remise en question, l'indépendance de **spécification** semble plus difficile à réaliser. Elle concerne avant tout notre seconde hypothèse.

VI.1.3.2. Spécification des fonctionnalités avant l'interface

La spécification de la seconde interface de notre application interactive nous a permis de constater que des informations qui devaient nécessairement provenir des fonctions (libellé_produit, nom_à_afficher...) n'avaient pas été définies. On peut alors parler d'erreur de spécification. Cependant, cette erreur a une explication : le fait de débuter par la spécification de l'application proprement dite a une incidence sur la qualité des informations qui seront offertes à l'utilisateur car elles ont été définies sans tenir réellement compte des besoins de ce dernier[1].

Nous avions signalé au chapitre I que notre méthodologie utilisait la tendance la plus ancienne qui était de débuter par la spécification des fonctions et de considérer l'interface comme un module suplémentaire d'entrée-sortie. Cette approche semble dès lors contraignante puisqu'elle ne centre pas suffisamment la spécification des fonctions sur leur utilisation.

Une solution est alors de réaliser la spécification de nos deux composants par itération : les spécifications des interfaces pourra entraîner une modification des spécifications fonctionnelles. Ceci renforce ce que nous avions dit à la section précédente : il ne peut et il ne **doit** pas y avoir d'indépendance entre la spécification de l'interface et des fonctionnalités.

VI.1.3.3. Indépendance de la Tâche par rapport à la Présentation

Nous avons posé l'hypothèse que la spécification de la Tâche se faisait indépendamment de toute implémentation qui en serait faite. Or, l'existence de certains messages est liée au style d'interaction qui sera utilisé. Plus précisément, nous pouvons faire la distinction entre un style d'interfaction offrant à l'utilisateur la possibilité de voyager parmi les messages de l'application et un style qui l'interdit. La lecture du schéma de la conversation se fera également de manière différente selon le choix qui sera fait.

L'indépendance n'est donc pas totale : le spécificateur de la Tâche doit connaître le degré de liberté offert à l'utilisateur : soit ce dernier choisira lui-même les messages qu'il veut traiter, soit l'interface lui présentera les alternatives qui lui sont offertes et procédera à son déplacement dans le schéma de la conversation.

VI.1.3.4. Définition précise de l'interface

La Tâche doit permettre de définir complètement et de manière simple l'interface qui sera réalisée. Cette hypothèse ne peut être vérifiée qu'en fonction de la spécification de la Présentation qu'elle permetra de définir et de l'implémentation qui en sera faite. Nous l'analyserons donc lors de l'évaluation des étapes suivantes.

VI.1.3.5. Conclusion

La spécification de la Tâche est une étape intermédiaire qui permet de traduire les éléments fonctionnels en termes significatifs pour l'utilisateur et d'y ajouter les informations qui lui seront offertes sans que les fonctions de l'application n'en aient connaissance.

---

[1] Le fait d'en tenir réellement compte présuppose une approche méthodologique différente de celle que nous avons utilisé. Cfr paragraphe suivant.

Le schéma de la conversation permet de définir l'enchaînement du dialogue qui devra être respecté. Pour évaluer complètement nos hypothèses, il nous faut encore passer aux étapes suivantes de notre méthodologie. C'est ce que nous proposons aux sections suivantes.

VI.3. Validation de la tâche par rapport à la dynamique des traitements

Cette phase consiste en la vérification de la correspondance entre les informations contenues dans les messages fonctionnels et les messages interactifs fonctionnels. Chaque information contenue dans un message fonctionnel doit se retrouver une et une seule fois dans un message interactif et vice-versa.

Cette correspondance est effectivement présente dans notre première interface par rapport aux spécifications fonctionnelles présentées au chapitre V. Nous avons relevé aux sections VI.1.2.2. et VI.1.3. qu'il était nécessaire de modifier la spécification des messages fonctionnels pour permettre à la seconde interface de disposer de tous les éléments qui doivent être communiqués au client. Cette modification doit alors être répercutée sur la première interface si l'on veut conserver l'indépendance modulaire entre les deux composantes de l'application (notre première hypothèse). Or la première interface ne manipule pas ces informations[1].

En toute généralité, les interfaces qui seront conçues pour une même application n'utiliseront pas toutes les mêmes informations : leur richesse dépendra de la classe d'utilisateur pour qui elles ont été créées ou de leur spécificité par rapport aux autres interfaces. Dans l'optique d'un tableau de bord permettant de renseigner l'utilisateur sur tous les éléments pertinents à la réalisation de sa tâche, le fait que l'utilisateur soit expert ou débutant aura une grande influence sur les informations qui devront être présentes. Tout ceci nous amène à dire que notre correspondance 1 à 1 entre les informations des messages fonctionnels et des messages interactifs fonctionnels nous semble trop forte puisque certaines informations fonctionnelles ne seront pas utilisées par toutes les interfaces. Un critère plus faible serait de dire :

Toute information dans un message interactif fonctionnel devra se retrouver dans un message fonctionnel.

Notre hypothèse d'indépendance entre les fonctionnalités et l'interface nous oblige ainsi à modifier quelque peu la phase de validation de la Tâche pour permettre la construction de plusieurs interfaces pour une même application.

---

[1] Nous prenons l'hypothèse que l'opérateur cherche avant tout à enregistrer un maximum de commandes plutôt qu'à veiller à leur validation. Ces informations ne l'intéresse donc pas. Le meilleur exemple d'informations inutiles est la limite d'achat qui figurera dans le message d'erreur Commande_non_valide de la seconde interface; il n'est d'aucune utilité pour l'opérateur tandis que le client pourra réduire sa commande en fonction de ce montant.

VI.4. Dynamique d'implémentation

VI.3.0. Introduction

Le schéma de la dynamique d'implémentation consiste en l'intégration de la dynamique des traitements au schéma de la conversation. Le schéma résultant permettra au concepteur d'analyser son application interactive en terme d'efficacité. Le déclenchement d'une fonction ne se fera en effet que lorsque toutes les informations dont elle a besoin ont été introduites. La rapidité d'exécution des fonctions dépendra dès lors de l'ordre dans lequel les données dont elles ont besoin seront saisies. Si le déclenchement de la première fonction de l'application proprement dite demande une information qui ne sera saisie par l'opérateur qu'en dernier lieu, l'exécution des fonctions ne débutera qu'après son introduction. Cela reste tout-à-fait admissible si l'utilisateur perçoit cette donnée comme la dernière à communiquer.

Le schéma de la dynamique d'implémentation est également utile à l'utilisateur car il lui indique le fonctionnement de l'application proprement dite. Il pourra de la sorte comprendre les mécanismes à la base de l'application interactive et améliorer sa connaissance liée à la tâche.

Son utilité est donc double. Il constitue un résumé à la fois des spécifications fonctionnelles (la dynamique des traitements) et de la Täche (le schéma de la conversation).

VI.3.1. Dynamique d'implémentation de l'interface pour une commande écrite

Voir figure VI.4.

VI.3.2. Dynamique d'implémentation de l'interface pour une commande téléphonée

Voir figure VI.5.

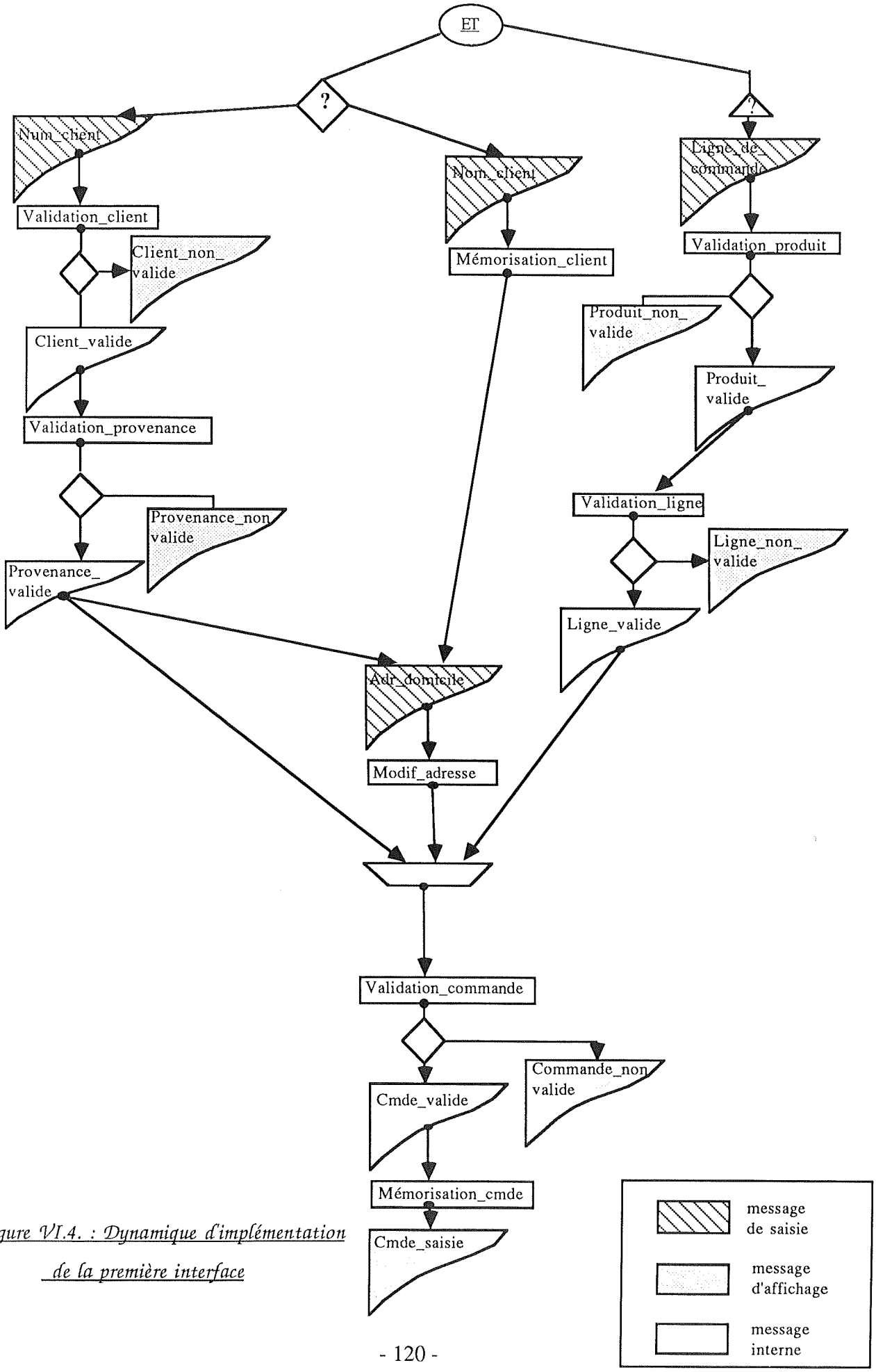


Figure VI.4. : Dynamique d'implémentation de la première interface

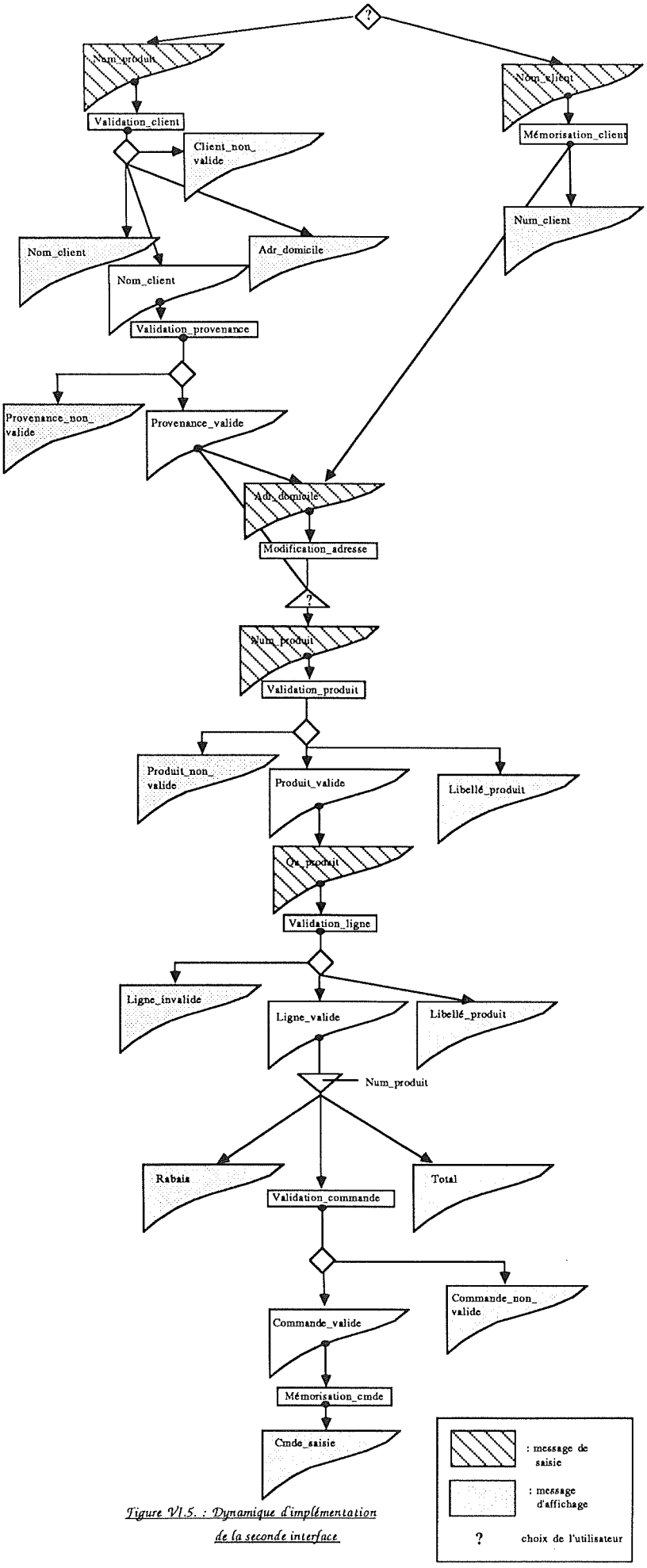


Figure VI.5. : Dynamique d'implémentation de la seconde interface

VI.4. Spécification de la Présentation

VI.4.0. Introduction

La présentation des messages interactifs à l'écran se fait à l'aide d'objets interactifs. L'utilisateur pourra effectuer sur ces objets des actions physiques qui seront la traduction des opérations sur les messages interactifs.

Un objet interactif dépendra directement du ou des styles d'interaction et de l'environnement qui seront utilisés. Par exemple, une fenêtre est un objet interactif qui ne sera pas disponible si le style d'interaction est le langage de commande; de surcroît, les caractéristiques d'une fenêtre varient fortement selon le type de machine ou le logiciel utilisés. En conséquence, la spécification de la Présentation devra se faire avec la connaissance du matériel (hardware ou software) qui sera employé pour réaliser l'application interactive.

Nous avons choisi de réaliser notre exemple sur deux environnements assez différents pour mieux analyser la portée de nos spécifications. Nous introduirons quelque peu ces deux logiciels avant de spécifier en ce qui les concerne les objets interactifs de notre saisie d'une commande. Le premier est le logiciel Hypercard tournant sur Macintosch. Il sera exposé à la section suivante. Le deuxième est Turbo/Pascal, un environnement de programmation sur compatibles IBM. Les spécifications qui le concernent sont présentées à l'annexe 2. Seules les conclusions qui s'y rapportent seront traitées dans ce chapitre.

VI.4.1. Spécifications pour Hypercard

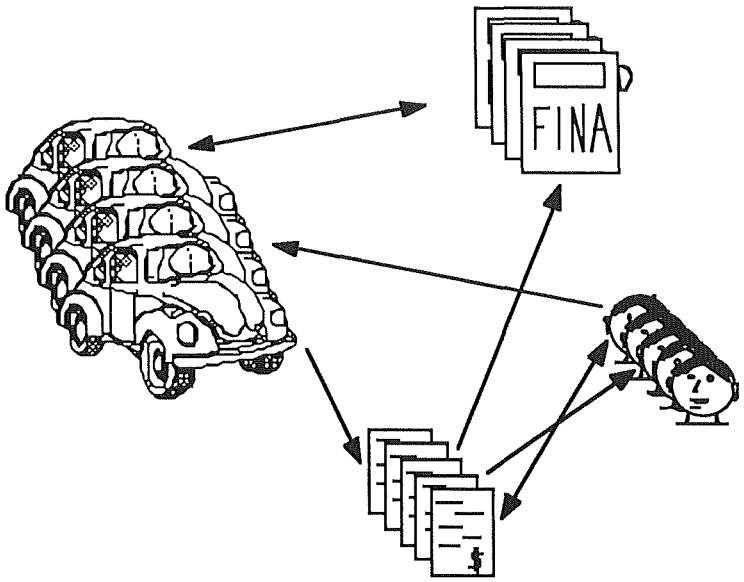
VI.4.1.0. Introduction à Hypercard

[Hypercard, 1987] et [Hypertalk, 1987]

1) Généralités

Hypercard est un logiciel permettant le stockage de données sous n'importe quelle forme (textuelle, graphique, sonore...)[1] en laissant la liberté la plus absolue à l'utilisateur quant à leur rangement. Pour cela, ce dernier définit des liens entre les données (un lien permet de voyager entre deux objets). Ces liens peuvent être statiques (le destinataire est constamment le même) ou dynamiques (ils se font par programme). On peut créer autant de liens par objet qu'on le désire (voir figure VI.6.). L'environnement est interactif : l'utilisateur créent ses objets et définit leurs liens en les manipulant.

[1] C'est ce que l'on désigne généralement par la notion d'hypertexte



.Figure VI.6. : Liens entre objets d'Hypercard

Derrière Hypercard, il existe un langage : Hypertalk. Il permet d'associer aux objets interactifs un ensemble de procédures qui s'exécuteront en réponse à un évènement bien précis (un click souris sur l'objet, la frappe de la touche <CR>...) déclenché par l'utilisateur, un autre objet ou le système d'exploitation (par communication de messages).

Grâce à ce langage, on peut créer des applications interactives dont le code sera disséminé parmi les objets interactifs de l'application[1].

Nous allons à présent examiner les objets interactifs standard d'Hypercard, ce qui nous permettra de préciser certaines notions brièvement abordées :

2) Les objets standard d'Hypercard

Les deux objets de bas niveaux d'Hypercard sont le bouton et le champ.

Un **bouton** est représenté à l'écran par une icône sous une forme quelconque (un rond, un rectangle, un dessin...). On ne peut pas stocker de l'information dans un bouton. On peut cependant lui associer des procédures qui seront exécutées en réponse aux actions de l'utilisateur sur cet objet. Par exemple, lorsque l'utilisateur cliquera sur un bouton à l'aide d'une souris, une action sera exécutée en réponse à cet événement. Cette action aura été préalablement programmée par le concepteur du dialogue.

Un **champ**, par contre, peut contenir de l'information. On pourra, de la même manière que le bouton, lui associer des procédures en réponse à des événements provoqués par l'utilisateur. La différence essentielle entre un bouton et un champ est lié au stockage d'information; les procédures associées à un champ auront la plupart du temps un rapport avec l'information contenue dans le champ tandis que les actions associées à un bouton peuvent être plus générales. Un champ sera dès lors tout-à-fait adapté pour présenter un message interactif.

[1] Hypercard utilise une approche orientée objet avec notion d'héritage.

La représentation courante d'un champ sera un rectangle[1] avec une ou plusieurs lignes dans lesquelles l'utilisateur pourra introduire de l'information.

Une **carte** est l'unité d'affichage à l'écran[2]. Il y aura à tout moment au maximum une carte affichée. Une carte pourra contenir un nombre quelconque de champs, de boutons ainsi que du texte, des graphiques. On pourra également lui associer des procédures.

Finalement, le **stack** est le regroupement logique d'un ensemble de cartes[3]. Des cartes de même nature seront regroupées dans un même stack.

Hypercard nous permettra non seulement de réaliser notre application interactive mais également de construire la base de données qui y sera attachée. Le logiciel n'est cependant pas à proprement parler un SGBD car les liens entre objets sont à la charge du concepteur. De plus, aucune règle ne doit être respectée dans la définition des liens[4].

Illustrons tout cela sur un exemple :

Dans notre exemple de saisie d'une commande, un produit est défini par les caractéristiques suivantes :

- un numéro de produit;
- un libellé;
- une unité-d'achat;
- un prix unitaire;
- un état d'approvisionnement;
- une date de réapprovisionnement;

L'ensemble des produits constituera le stack Produits. Chaque carte du stack représentera une occurence d'un produit.

Une carte produit sera constituée :

- des champs num_produit, libellé, unité_d'achat, prix_unitaire, état_d'appro et date_de_réappro;

- de texte explicatif à côté de ces champs;

- des boutons home (retour au menu principal), client (pour accéder au stack des clients), commande (pour accéder au stack des commandes saisies), substitution (pour accéder au produit de substitution s'il y en a), avant et après (pour accéder respectivement au produit précédent et suivant).

Une carte Produit est présentée à la figure VI.7.

---

[1] Le rectangle pourra être ombré, arrondi, avec un cadre invisible, des lignes invisibles...

[2] Une carte est un type de fenêtre spécifique à Hypercard.

[3] Un stack est comparable à un fichier de cartes.

[4] Voir annexe 2.

Les objets interactifs standard d'Hypercard constituent des classes génériques d'objet. Un bouton pourra par exemple être représenté sous forme d'icône, de rectangle...Les objets interactifs visualisant les messages interactifs que nous avons spécifiés seront des occurences des types que nous venons de présenter[1]. Nous spécifions à la section suivante la Présentation de la première interface.

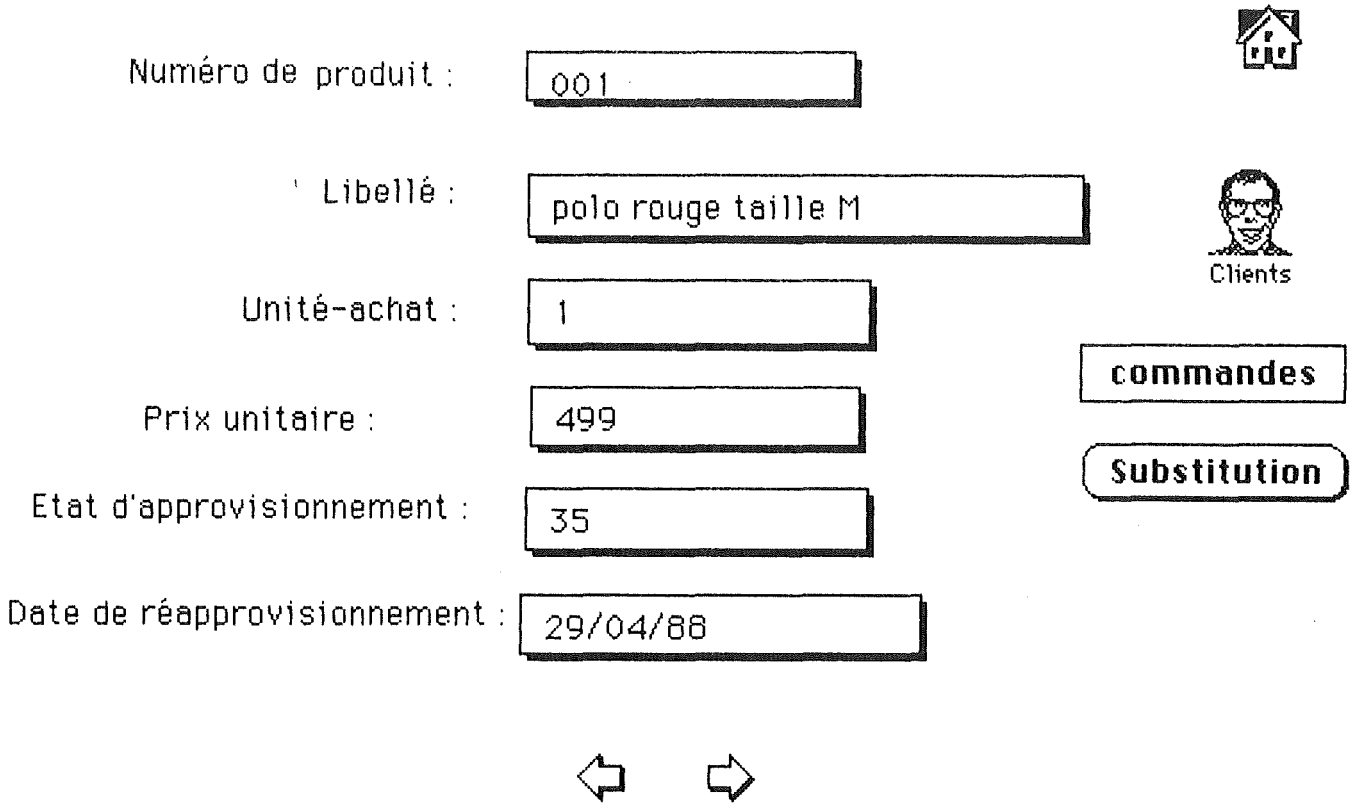


Figure VI.7. : Une carte Produit

VI.4.1.1. Spécification des objets interactifs pour une commande écrite

* Num_client :

Nom: num_client;
Type : un champ;
Contenu ou visualisation : un rectangle d'une ligne et un libellé à sa gauche : "Numéro du client";
Justification : objet standard permettant la saisie d'informations;
Message interactif qu'il représente : message num_client_à_saisir;
Opérations portant sur le contenu sémantique :

- sélectionner le message --> cliquer sur l'objet;
- affecter une valeur --> taper au clavier une valeur après avoir sélectionné l'objet;
- corriger une valeur --> utiliser la touche <delete> ou utiliser la souris en sélectionnant la partie à corriger et retaper la nouvelle valeur (décomposition des actions d'effacement et d'entrée d'une valeur). L'objet doit avoir été préalablement sélectionné;

[1] Plus précisément des spécialisations de ces types puisqu'ils possèderont toutes les caractéristiques de leur classe générique mais avec des éléments supplémentaires.

- effacement d'une valeur --> utiliser la touche <delete> en supprimant l'entièreté de la valeur ou sélectionner l'entièreté de la valeur à l'aide de la souris et taper sur la touche <delete>;
- clôturer le message --> déplacer le curseur ou cliquer sur tout autre objet;

Contraintes d'enchaînement : num_client ne peut être saisi que si nom_client ne l'a pas été préalablement;
Fonctions déclenchées : validation_client, validation_provenance;

* Nom_client :

Nom : Nom_client;
Type : un champ;
Contenu ou visualisation : un rectangle avec trois lignes (l'une pour le nom, l'une pour le prénom et la dernière pour le titre) avec un libellé explicatif à la gauche de chaque ligne;
Justification : objet standard permettant la saisie d'informations;
Message interactif qu'il représente : message nom_client_à_saisir;
Opérations portant sur le contenu sémantique :

- sélectionner le message --> cliquer sur l'objet;
- affecter une valeur à un attribut --> taper au clavier une valeur après avoir sélectionné la ligne de l'attribut;
- corriger une valeur d'un attribut --> utiliser la touche <delete> ou utiliser la souris en sélectionnant la partie à corriger et retaper la nouvelle valeur (décomposition des actions d'effacement et d'entrée d'une valeur). La ligne de l'attribut doit avoir été préalablement sélectionnée;
- effacement d'une valeur --> utiliser la touche <delete> en supprimant l'entièreté de la valeur ou sélectionner l'entièreté de la valeur à l'aide de la souris et taper sur la touche <delete>;
- clôturer le message --> déplacer le curseur ou cliquer sur tout autre objet;

Contraintes d'enchaînement : le nom du client ne peut être saisi que si le numéro ne l'a pas été préalablement;
Fonctions déclenchées : mémorisation_client;

* Adr_domicile :

Nom : Adr_domicile;
Type : un champ;
Contenu ou visualisation : un rectangle avec deux lignes (l'une pour la rue et le numéro, l'autre pour le NPA et la localité) et un libellé explicatif à la gauche de chaque ligne;
Justification : objet standard permettant la saisie d'informations;
Message interactif qu'il représente : message adr_domicile_à_saisir.
Opérations portant sur le contenu sémantique :

- sélectionner le message --> cliquer sur l'objet;
- affecter une valeur à un attribut --> taper au clavier une valeur après avoir sélectionné la ligne de l'attribut;
- corriger une valeur d'un attribut --> utiliser la touche <delete> ou utiliser la souris en sélectionnant la partie à corriger et retaper la nouvelle valeur (décomposition des actions d'effacement et d'entrée d'une valeur). La position de l'attribut doit avoir été préalablement sélectionnée;

- effacement d'une valeur d'un attribut --> utiliser la touche <delete> en supprimant l'entièreté de la valeur ou sélectionner l'entièreté de la valeur à l'aide de la souris et taper sur la touche <delete>;
- clôturer le message --> déplacer le curseur ou cliquer sur tout autre objet;

Contraintes d'enchaînement : l'adresse du client ne peut être saisie que si le numéro du client ou son nom a été préalablement saisi; sa saisie est obligatoire si le nom du client a été saisi;
Fonctions déclenchées : modification_adresse;

*Lignes_de_commande :

Nom: lignes_de_commande;
Type : un champ;
Contenu ou visualisation : un rectangle avec une scroll-bar, un libellé à la gauche et autant de lignes que l'utilisateur le désire. Une ligne contient un numéro de produit et une quantité (facultative);
Justification : objet standard permettant la saisie d'informations;
Messages interactifs qu'il représente : l'ensemble des messages ligne_commande;
Opérations portant sur le contenu sémantique :

- sélectionner un message --> cliquer sur l'objet;
- affecter une valeur à un attribut --> taper au clavier une valeur après avoir sélectionné la ligne de l'attribut; on impose que la première valeur introduite soit le numéro du produit et la deuxième la quantité (facultative) séparée par un blanc. S'il y a plus de deux valeurs séparées par un blanc, on ne tient pas compte de ce qui suit la deuxième valeur;
- corriger la valeur d'un attribut --> utiliser la touche <delete> ou utiliser la souris en sélectionnant la partie à corriger et retaper la nouvelle valeur (décomposition des actions d'effacement et d'entrée d'une valeur). La partie à corriger doit avoir été préalablement sélectionnée;
- effacement de la valeur d'un attribut --> utiliser la touche <delete> en supprimant l'entièreté de la valeur ou sélectionner l'entièreté de la valeur à l'aide de la souris et taper sur la touche <delete>;
- clôturer un message --> déplacer le curseur ou cliquer sur tout autre objet;

Contraintes d'enchaînement : aucune;
Fonctions déclenchées : validation_produit et validation_ligne;

* OK :

Nom : ok;
Type : un bouton;
Contenu ou visualisation : un rectangle arrondi avec le libellé "OK";
Justification : objet standard permettant le déclenchement d'une action (ici l'action est le signalement de la fin de la saisie des lignes de la commande);
Message interactif qu'il représente : message saisie_terminée;
Opérations portant sur le contenu sémantique :

- clôturer le message --> cliquer sur l'objet;

Contraintes d'enchaînement : il doit y avoir eu une saisie préalable du nom ou du numéro du client ainsi que la saisie d'au moins une ligne valide;
Fonctions déclenchées: validation_commande, mémorisation_commande;

* Annuler :

Nom : annuler;
Type : un bouton;
Contenu ou visualisation : un rectangle arrondi avec le libellé "Annuler";
Justification : objet standard permettant le déclenchement d'une action (ici l'action est l'annulation de la commande);
Message interactif qu'il représente : message saisie_annulée;
Opérations portant sur le contenu sémantique :
- clôturer le message --> cliquer sur l'objet;

Contraintes d'enchaînement : aucune;
Fonctions déclenchées : aucune;

* Message_aide :

Nom : Message_aide;
Type : une carte;
Contenu ou visualisation : un texte correspondant au contenu du message interactif aide_phase;
Justification : objet standard permettant l'affichage d'un ensemble d'informations (ici, un message d'aide);
Message interactif qu'il représente : message aide_sémantique;
Opérations portant sur le contenu sémantique :
- ranger le message --> cliquer sur l'objet;
- sélectionner le message --> cliquer sur l'icône représentant le message lorsqu'il n'est pas sélectionné;

Contraintes d'enchaînement : à n'importe quel moment;
Fonctions déclenchées : aucune;

* Client-invalide :

Nom : Client_invalide;
Type : une carte;
Contenu ou visualisation : le texte de la définition du message interactif correspondant;
Justification : objet permettant l'affichage d'un ensemble d'informations (ici, d'un message d'erreur);
Message interactif qu'il représente : client_non_valide;
Opérations portant sur le contenu sémantique :
- ranger le message --> cliquer sur la carte;

Contraintes d'enchaînement : provient de l'application lorsque l'erreur correspondante est survenue;
Fonctions déclenchées : aucune;

* Provenance_invalide :

Nom : Provenance_invalide;
Type : une carte;
Contenu ou visualisation : le texte de la définition du message interactif correspondant;
Justification : objet permettant l'affichage d'un ensemble d'informations (ici, d'un message d'erreur);

Message interactif qu'il représente : provenance_non_valide;
Opérations portant sur le contenu sémantique :
- ranger le message --> cliquer sur la carte;

Contraintes d'enchaînement : provient de l'application lorsque l'erreur correspondante est survenue;
Fonctions déclenchées : aucune;

* Produit_invalide :

Nom : Produite_invalide;
Type : une carte;
Contenu ou visualisation : le texte de la définition du message interactif correspondant;
Justification : objet permettant l'affichage d'un ensemble d'informations (ici, d'un message d'erreur);
Message interactif qu'il représente : produit_non_valide;
Opérations portant sur le contenu sémantique :
- ranger le message --> cliquer sur la carte;

Contraintes d'enchaînement : provient de l'application lorsque l'erreur correspondante est survenue;
Fonctions déclenchées : aucune;

* Ligne_invalide :

Nom : Ligne_invalide;
Type : une carte;
Contenu ou visualisation : le texte de la définition du message interactif correspondant;
Justification : objet permettant l'affichage d'un ensemble d'informations (ici, d'un message d'erreur);
Message interactif qu'il représente : ligne_non_valide;
Opérations portant sur le contenu sémantique :
- ranger le message --> cliquer sur la carte;

Contraintes d'enchaînement : provient de l'application lorsque l'erreur correspondante est survenue;
Fonctions déclenchées : aucune;

* Commande_invalide :

Nom : Commande_invalide;
Type : une carte;
Contenu ou visualisation : le texte de la définition du message interactif correspondant;
Justification : objet permettant l'affichage d'un ensemble d'informations (ici, d'un message d'erreur);
Message interactif qu'il représente : commande_non_valide;
Opérations portant sur le contenu sémantique :
- ranger le message --> cliquer sur la carte;

Contraintes d'enchaînement : provient de l'application lorsque l'erreur correspondante est survenue;
Fonctions déclenchées : aucune;

* Num_client_entier :

Nom : Num_client_entier;
Type : une boîte de dialogue;
Contenu ou visualisation : le texte de la définition du message interactif correspondant;
Justification : permet l'affichage d'un texte à l'utilisateur;
Message interactif qu'il représente : num_client_entier;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur l'objet;

Contraintes d'enchaînement : affiché par l'interface lorsque l'erreur correspondante est survenue;
Fonctions déclenchées : aucune;

* NPA_entier :

Nom : NPA_entier;
Type : une boîte de dialogue;
Contenu ou visualisation : le texte de la définition du message interactif correspondant;
Justification : permet l'affichage d'un texte à l'utilisateur;
Message interactif qu'il représente : adr_dom_NPA_entier;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur l'objet;

Contraintes d'enchaînement : affiché par l'interface lorsque l'erreur correspondante est survenue;
Fonctions déclenchées : aucune;

* Saisie_ref_client :

Nom : saisie_ref_client;
Type : une boîte de dialogue;
Contenu ou visualisation : le texte de la définition du message interactif correspondant;
Justification : permet l'affichage d'un texte à l'utilisateur;
Message interactif qu'il représente : saisie_référence_client;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur l'objet;

Contraintes d'enchaînement : affiché par l'interface lorsque l'erreur correspondante est survenue;
Fonctions déclenchées : aucune;

* Quantité_entière_positive :

Nom : quantité_entière_positive;
Type : une boîte de dialogue;
Contenu ou visualisation : le texte de la définition du message interactif correspondant;
Justification : permet l'affichage d'un texte à l'utilisateur;
Message interactif qu'il représente : quantité_entière_positive;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur l'objet;

Contraintes d'enchaînement : affiché par l'interface lorsque l'erreur correspondante est survenue;

Fonctions déclenchées : aucune;

* Num_prod_incorrect :

Nom : Num_prod_incorrect;
Type : une boîte de dialogue;
Contenu ou visualisation : le texte de la définition du message interactif correspondant;
Justification : permet l'affichage d'un texte à l'utilisateur;
Message interactif qu'il représente : Num_produit_incorrect;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur l'objet;

Contraintes d'enchaînement : affiché par l'interface lorsque l'erreur correspondante est survenue;
Fonctions déclenchées : aucune;

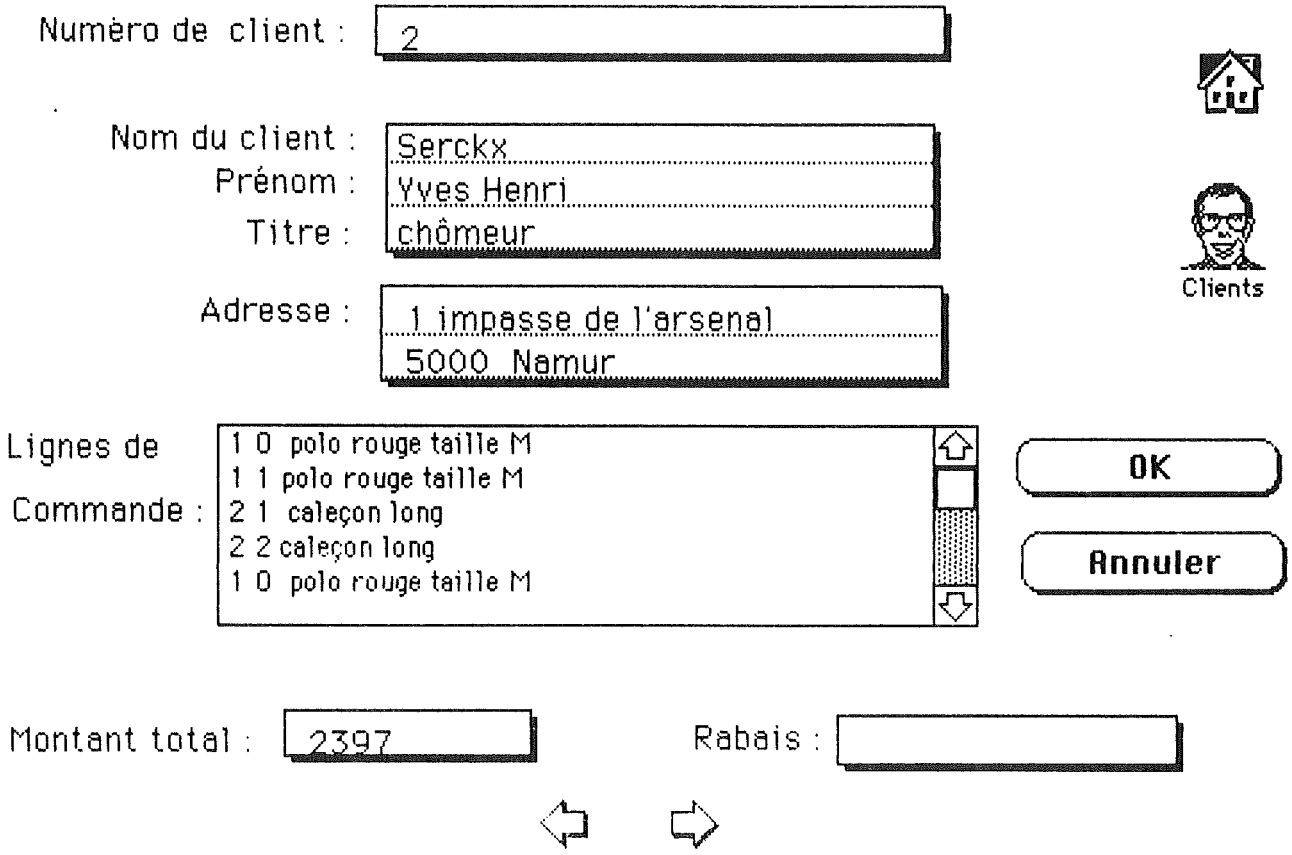


Figure VI.8. : commande enregistrée

* Commande_saisie :

Nom : commande saisie;
Type : une carte;
Contenu ou visualisation : la carte est constituée des champs numéro_de_commande, date_commande, montant_total, rabais, num_client, adr_domicile et lignes_commande. Tous ces champs sont d'affichage. La carte est également constituée des boutons cmde_avant (aller à la commande_saisie précédente), cmde_après (aller à la commande suivante), Home (retour au menu principal), go_clients (pour accéder au stack des clients), go-produits (pour accéder au stack des produits). Voir figure VI.9.;
Justification : permet l'affichage d'un ensemble d'informations;

Message interactif qu'il représente : commande_enregistrée;
Opérations portant sur le contenu sémantique :
- ranger le message --> cliquer sur les boutons Home, go-clients ou go-produits;
- sélectionner le message --> utiliser les objets permettant de voyager entre cartes;

Contraintes d'enchaînement : aucune;

Les objets interactifs correspondant à des messages fonctionnels interactifs de saisie seront regroupés dans une carte Commande_à_saisir qui sera similaire au bon de commande à partir duquel l'opérateur enregistre la commande.

VI.4.1.2. Remarques sur la spécification des objets interactifs de la première interface

a) Correspondance avec les messages interactifs

Certains messages purement interactifs qui avaient été spécifiés lors de l'analyse de la Tâche n'ont pas de correspondance dans la Présentation. Ils s'agit des messages Quitter_saisie et Choix_saisie :

- Quitter_saisie était le message par lequel l'utilisateur signalait à l'interface son désir de quitter l'application interactive. Hypercard possède un item de menu ("Quit") présentant ce message et évitant de le créer (sa correspondance existe mais ne doit pas être spécifiée).

- Choix_saisie permettait à l'utilisateur de choisir dans la structure parallèle le message de saisie qu'il désirait traiter. Ce message n'est utile que dans un environnement ne permettant pas d'effectuer ce choix par une sélection du message à saisir (cfr section VI.1.3.). Nous n'en avons donc pas besoin pour notre environnement.

b) Le rangement et la suppression

L'analyse de la tâche ne pose pas d'hypothèse sur l'implémentation qui sera faite de l'application interactive[1]. Les messages d'affichage que nous avions définis pouvaient être rangés ou supprimés par l'utilisateur selon la gestion qui en serait faite. Nous avons donc dû faire des choix :

- les objets interactifs présentant les messages d'erreur sémantique et le message d'aide seront mémorisés dans les stacks prévus à cet effet (un stack pour les erreurs sémantiques et un autre pour l'aide). L'affichage de ces messages se fera en suivant le lien statique défini entre la carte Commande_à_saisir (présentant le bon de commande) et la carte présentant le message. Le rangement de celui-ci se fera en suivant le chemin inverse. L'utilisateur pourra accéder à ces cartes en dehors de l'application interactive et les adapter à ses besoins.

- les objets interactifs présentant les messages d'erreur syntaxique seront créés dynamiquement par l'interface lorsque l'erreur correspondante est survenue. Ils consistent en une boîte de dialogue dans laquelle on affiche le texte désiré. L'utilisateur la supprimera en cliquant dessus.

---

[1] Cfr cependant l'évaluation de la Tâche.

Nous avons relevé à la section IV.1.4.2. l'utilité de disposer de messages interactifs enregistrés sur un support secondaire et accessibles par l'utilisateur pour pouvoir les adapter. Nous avons utilisés ici les deux approches pour illustrer notre propos.

c) La sélection et la clôture

Les messages purement interactifs Saisie_terminée et Saisie_annulée pouvaient être manipulés par les primitives de sélection et de clôture. Le fait que la visualisation de ces deux messages est présente en permanence (les objets interactifs OK et Annuler sont affichés continuellement) évite de devoir les sélectionner pour pouvoir directement les clôturer. Rappelons en effet que ces messages ne contiennent pas d'informations à saisir mais possèdent une sémantique implicite (leur clôture provoque leur envoi à l'interface qui connait leur signification).

d) La synchronisation

Nous avons dit à la section VI.1.1.5. qu'une synchronisation pouvait être de nature différente (automatique ou explicite). Le bouton OK représente l'objet sur lequel l'opérateur cliquera lorsqu'il estimera avoir terminé la saisie de la commande. L'implémentation de cet objet pourra conduire à deux situations diffférentes :

- Soit l'interface gère les conditions de la synchronisation (la saisie des références du client et au moins une ligne de commande valide). Elle sait dès lors quand l'utilisateur pourra cliquer sur l'objet. Elle pourra le présenter dès que les conditions sont respectées ou l'afficher en permanence et empêcher toute action sur l'objet tant qu'elles ne sont pas respectées. Aucune erreur ne peut alors être commise car l'utilisateur manipulera l'objet lorsque les conditions sont respectées.

- Soit l'interface se contente d'attendre que l'utilisateur clique sur l'objet pour vérifier ensuite si les conditions de la synchronisation sont effectivement vérifiées. Si elles ne le sont pas, des messages d'erreur devront être affichées.

De l'implémentation qui sera faite de la synchronisation dépendra l'utilité de l'objet Saisie_ref_client. Nous l'avons spécifié en toute généralité mais .

Dans un environnement plus traditionnel (voir annexe 2 et section VI.4.3.1.) la synchronisation sera représentée par une commande au lieu d'un objet interactif.

VI.4.1.3. Spécification des objets interactifs pour une commande téléphonée

Nous ne spécifierons à nouveau que les objets qui différent de la première interface. Nous avons choisi d'offrir à l'utilisateur une présentation de l'application interactive similaire à la première interface pour bénéficier d'une uniformité de conception.

* Num_client

Messages interactifs qu'il représente : messages Num_client_à_saisir et Num_client_à_afficher;

* Nom_client

Messages interactifs qu'il représente : messages Nom_client_à_saisir et Nom_client_à_afficher;

* Adresse_domicile

Messages interactifs qu'il représente : les messages Adr_domicile_à_afficher et Adr_domicile_à_saisir;

* Lignes_de_commande

Nom: lignes_de_commande;
Type : un champ;
Contenu ou visualisation : un rectangle avec une scroll-bar, un libellé à la gauche et autant de lignes que l'utilisateur le désire. Une ligne contientra un numéro de produit, un libellé et une quantité (facultative);
Justification : objet standard permettant la saisie d'informations;
Messages interactifs qu'il représente : les messages Num_produit, Qu_produit et Libellé_produit;
Opérations portant sur le contenu sémantique :

- sélectionner un message --> cliquer sur l'objet;
- affecter une valeur à un attribut (le produit ou la quantité)--> taper au clavier une valeur après avoir sélectionné la ligne de l'attribut; on impose que la première valeur introduite soit le numéro du produit et la deuxième la quantité (facultative) séparée par un blanc. S'il y a plus de deux valeurs séparées par un blanc, on ne tient pas compte de ce qui suit la deuxième valeur;
- corriger la valeur d'un attribut (le produit ou la quantité) --> utiliser la touche <delete> ou utiliser la souris en sélectionnant la partie à corriger et retaper la nouvelle valeur (décomposition des actions d'effacement et d'entrée d'une valeur). La partie à corriger doit avoir été préalablement sélectionnée;
- effacement de la valeur d'un attribut (le produit ou la quantité) --> utiliser la touche <delete> en supprimant l'entièreté de la valeur ou sélectionner l'entièreté de la valeur à l'aide de la souris et taper sur la touche <delete>;
- clôturer un message --> déplacer le curseur ou cliquer sur tout autre objet;

Contraintes d'enchaînement : aucune;
Fonctions déclenchées : validation_produit et validation_ligne;

* Montant_total :

Nom : montant_total;
Type : un champ;
Contenu ou visualisation : un rectangle avec une ligne dans laquelle apparaîtra le montant de la commande et un libellé explicatif à sa gauche;
Justification : objet permettant l'affichage d'un ensemble d'informations;

Message interactif qu'il représente : message total;
Opérations portant sur le contenu sémantique :
- L'objet représente un message interactif d'affichage; il n'y a pas d'opération associée à l'objet car l'interface est seule à le gérer;

Contraintes d'enchaînement : il prendra une valeur lorsque la commande aura été validée;
Fonctions déclenchées : aucune;

* Rabais :

Nom : rabais;
Type : un champ;
Contenu ou visualisation : un rectangle avec une ligne dans laquelle apparaîtra le rabais possible de la commande;
Justification : objet permettant l'affichage d'un ensemble d'informations;
Message interactif qu'il représente : message rabais;
Opérations portant sur le contenu sémantique :
- L'objet représente un message interactif d'affichage; il n'y a pas d'opération associée à l'objet car l'interface est seule à le gérer;

Contraintes d'enchaînement : il prendra une valeur lorsque la commande aura été validée et si le client est membre du personnel;
Fonctions déclenchées : aucune;

VI.4.1.4. Remarques sur la spécification de la seconde interface

Un même objet interactif pourra présenter à l'écran des messages de types différents possédant la même structure de données (Num_client représente les messages Num_client_à_saisir et Num_client_à_afficher...) ou non (Ligne_de_commande regroupe Num_produit, Qu_produit et Libellé_produit).

Les objets interactifs supplémentaires sont nécessaires pour présenter les informations inexistantes dans la première interface (Montant_total, Rabais...). Notons que l'objet Lignes_commande possède une structure de données plus riche que pour la première interface car il comprendra également le libellé des produits qui seront commandés.

VI.4.1.5. Conclusion sur les spécifications conçues pour Hypercard

La spécification des objets interactifs de notre phase s'est basée sur les objets standard offerts par hypercard et l'analyse de la Tâche. Les contraintes sur la présentation ont influencés le choix de ces objets[1].

Regroupement et décomposition des messages

Un objet interactif pourra présenter plusieurs messages interactifs à l'écran. Dans le cas de regroupement de messages contenant les mêmes informations mais de types différents (Num_client_à_saisir et Num_client_à_afficher...) il est naturel d'utiliser un seul et même objet.

Lorsque les messages sont différents (Ligne_de_commande présente Qu_produit, Num_produit et Libellé_produit) cela reste en accord avec la définition d'un message interactif (une unité de saisie ou d'affichage pour l'utilisateur) si le regroupement permet de dissocier les

[1] C'était d'ailleurs là leur objectif.

différentes unités qui le constituent. Les objets interactifs doivent être **transparents** : les messages interactifs qu'ils visualisent doivent pouvoir être répertoriés[1].

Une attention toute particulière devra dès lors être apportée à la transposition des messages en objets interactifs pour permettre à l'utilisateur de distinguer parmi ces derniers les unités de dialogue qui lui sont offertes.

La visualisation d'une opération par un objet

Le message d'aide que nous avons défini sera représenté par une carte qui apparaîtra lorsque l'utilisateur l'aura sélectionnée. Cependant, il est nécessaire d'offrir un élément de présentation pour que l'utilisateur puisse le sélectionner (par exemple une icône ou un élément de menu). L'opération de sélection sera dès lors visualisée par un objet interactif[2].

Validation de la Présentation par rapport à la tâche

Nous avons défini une phase de validation de la tâche par rapport aux traitements. Il nous semble tout aussi raisonnable d'effectuer la même opération pour vérifier que tous les éléments des messages interactifs soient présentés à l'utilisateur. Néanmoins, nous avons fait remarqué (cfr section VI.4.1.2.) que l'existence de certains messages interactifs dépendait de l'implémentation qui serait réalisée. La correspondance devra dès lors se faire en tenant compte des choix qui ont été faits. La phase de spécification des objets interactifs sera suivie de cette validation.

Conclusion

Le choix des objets qui seront présentés à l'opérateur ont une influence considérable sur l'implémentation qui sera faite de l'application interactive. Ils devront respecter les modes de pensée de l'utilisateur et ne pas cacher la décomposition des messages interactifs. Nous avons voulu mettre en lumière plusieurs alternatives qui pouvaient s'offrir au concepteur. L'existence de certains objets dépendra de l'alternative choisie (cfr section VI.4.1.2.).

VI.4.2. Evaluation de l'implémentation réalisée à l'aide d'Hypercard

L'annexe 3 présente l'implémentation qui a été réalisée pour la seconde interface. Nous reprenons dans cette section une remarque qui concerne le schéma de la conversation :

L'implémentation du schéma de la conversation a été réalisée en restreignant les possibilités de l'utilisateur. La manipulation directe lui permettait en effet de saisir les informations nécessaires aux fonctions dans l'ordre qu'il désirait. Il a alors fallu lui interdire certaines actions pour respecter l'enchaînement défini.

Dans de telles conditions, on peut s'interroger sur l'utilité d'un schéma de la conversation. Le fait d'obliger l'utilisateur a saisir préalablement le nom d'un nouveau clientavant l'introduction de son adresse peut se comprendre d'un point de vue logique mais rien ne dit que cette logique sera celle de l'utilisateur!

---

[1] Par exemple, un opérateur voyant l'objet Ligne_de_commande devra y voir la représentation d'un ensemble de lignes contenant un numéro de produit, une quantité et un libellé. La Présentation devra permettre de dissocier ces différents messages.

[2] L'annexe 2 montrera que le contraire est également possible : un objet pourra être représenté par une action.

Nous pensons toutefois qu'il ne doit pas être remis en question car il existe des situations ou une certaine contrainte doit être spécifiée selon les caractéristiques de la tâche[1] ou de l'utilisateur[2].

VI.4.3. Spécifications pour Turbo/Pascal

Les spécifications réalisées pour l'environnement Turbo/Pascal sont présentées à l'annexe 2. Nous exposons ici les remarques qui en découlent.

VI.4.3.1. Présentation d'un message interactif par une action

Les messages Saisie_annulée Saisie_terminée et Quitter_saisie seront présentés à l'opérateur sous forme d'actions. Ces messages n'ont pas de contenu informatif mais possèdent une sémantique implicite (cfr VI.4.1.2.c). Leur présentation peut alors très bien se faire par une commande. Ces messages sont tous trois de contrôle; ils sont en fait la représentation d'une opération que l'utilisateur peut effectuer. Il est dès lors normal qu'ils puissent apparaître sous forme d'actions.

VI.4.3.2. Messages interactifs superflus

Le message Saisie_ref_client est inutile pour nos spécifications car ce type d'erreur ne pourra pas survenir. L'utilisateur sera obligé de débuter l'application par les références du client.

Les messages Num_client_entier, NPA_entier et Qu_entière_positive sont également inutiles car l'interface gèrera les caractères introduits par l'opérateur. Ceci est un choix d'implémentation qui devra bien sûr être communiqué à l'implémenteur.

VI.4.3.3. Message interactif supplémentaire

Lorsque l'opérateur a terminé la saisie du numéro d'un client, il a la possibilité de modifier son adresse ou de passer directement à la saisie des numéros de lignes. Nous avons donc omis de spécifier un message de contrôle lui offrant cette possibilité. Ce message ne sera à nouveau nécessaire que dans un environnement ne permettant pas de voyager entre les objets de l'application.

VI.4.3.4. Sélection des attributs d'un message

Nos spécifications ne permettent pas de sélectionner un attribut d'un message. L'opérateur introduira une valeur lorsque l'interface aura positionné le curseur sur l'attribut à saisir. L'opération de sélection d'un attribut n'est en effet possible que lorsqu'un mécanisme est offert pour se déplacer parmi les éléments d'un ou plusieurs objets. Il nous eût été possible de permettre cette sélection en fournissant à l'opérateur une fonction de naviguation parmi les

---

[1] Un haut degré de performance sera plus rapidement atteint si l'utilisateur doit employer constamment la même suite d'action; il la mémorisera au bout d'un certain temps et ne devra pas perdre de temps à choisir parmi les alternatives qui lui sont offertes.

[2] Un novice sera content d'être dirigé par l'application.

attributs (par exemple la touche <TAB>). Nos spécifications devenaient alors semblables à l'environnement Hypercard.

VI.4.3.4. Conclusion

Les spécifications qui ont été élaborées pour l'environnement Turbo/Pascal diffèrent sensiblement de celles d'Hypercard. L'élément majeur à la base de ces différences est la possibilité ou non de voyager parmi les attributs des messages ou entre les messages.

Nous avons également souligné qu'un message interactif pouvait être traduit au niveau de la Présentation par une action (une commande) de l'opérateur. Le message ne contient alors pas d'information manipulable et consiste en un renseignement ("je désirerais quitter l'application", "j'ai terminé la saisie des lignes de la commande"...) que l'utilisateur communique à l'interface. Ce renseignement peut être compris comme une opération ne portant pas sur un message bien précis mais sur un ensemble de messages[1].

La phase de spécification des objets interactifs pose des choix sur l'implémentation qui sera faite. L'analyse de la Tâche se devait d'être indépendante de l'environnement ou de l'implémentation. Les décisions qui seront prises dans la spécification de la Présentation auront une influence sur les messages qui seront réellement présentés.

VI.5. Evaluation de la méthodologie

Cette section reprend les diverses remarques que nous avons pu faire tout au long de ce chapitre. Nous les regroupons en fonction de nos hypothèses de départ :

VI.5.1. Indépendance des fonctionnalités par rapport à l'interface

L'indépendance **modulaire** est nécessaire à la conception de plusieurs interfaces pour une même application. Le code de l'application proprement dite est alors utilisé par les différents modules constituant les différentes interfaces. Cette forme d'indépendance constitue le fondement de notre méthodologie et ne doit pas être remise en question.

L'indépendance de **spécification** des deux composantes est par contre indésirable : les informations qu'elles se communiqueront pourront être nécessaires aux fonctionnalités ou à l'interface. Chacun définira ses besoins et leur réunion constituera les messages fonctionnels qui devront être définis.

Les différentes interfaces qui seront spécifiées n'auront pas toutes les mêmes besoins en informations. Ceci peut poser un problème, par exemple si une interface est ajoutée à une application interactive existante qui ne lui fournit pas les informations dont elle a besoin. L'indépendance des deux composantes est alors sérieusement compromise puisqu'il faudra modifier les fonctions de l'application (pour qu'elle communique les informations manquantes) en vue de conserver leur séparation modulaire. **Il est dès lors nécessaire de fournir aux interfaces une copie complète des informations que manipulent les fonctions de l'application pour garantir l'indépendance modulaire.**

---

[1] Par exemple, le message Saisie_terminée porte sur l'ensemble des informations de la commande à saisir.

En conclusion, il est difficile d'assurer une parfaite indépendance entre les deux parties d'une application interactive par le simple fait qu'elles sont étroitement liées : les fonctions ont besoin de l'interface pour communiquer avec l'environnement tandis que l'interface a besoin de celles-ci pour répondre aux demandes de l'utilisateur.

VI.5.2. Spécifications fonctionnelles avant le dialogue

Le fait de débuter les spécifications de l'application interactive par les fonctionnalités peut entraîner une incomplétude dans les informations qui seront fournies aux différentes interfaces. Elles ne tiennent en effet pas suffisamment compte des besoins de l'utilisateur pour être à même de lui fournir toutes les informations dont il a besoin. Ceci suggère de commencer par la spécification des interfaces à réaliser[1].

Un compromis nous semble judicieux : les deux spécifications seront définies en parallèle et les informations qu'elles nécessitent seront ensuite confrontées. Chacune de ces informations devra alors se retrouver dans un message fonctionnel. Nous avons précisé à la section VI.4. que chaque interface choisirait alors parmi ces informations celles dont elle a besoin.

VI.5.3. Indépendance de la Tâche par rapport à l'implémentation

Nous avons relevé lors des sections précédentes que la spécification de la Tâche dépendait du style d'interaction et du type de synchronisation qui seraient utilisés. Cependant, cette dépendance n'existera que si l'on veut que chaque information définie lors de cette phase se retrouve dans la spécification de la Présentation.

L'indépendance de la Tâche par rapport à la Présentation reste tout-à-fait réalisable si elle se fait sans poser de choix : on spécifie tous les messages et les opérations qui peuvent exister, quelque soit la Présentation qui en sera faite. C'est en fait la méthode que nous avons suivie pour la spécification de notre exemple. La conséquence de cette indépendance est l'inutilité de certains messages : ce sera lors de la définition de la Présentation que ces choix seront faits.

VI.5.4. Complétude et simplicité de la spécification de la Tâche

Les messages interactifs et les opérations à l'aide desquelles l'utilisateur pourra les manipuler constituent les informations nécessaires à la spécification des objets interactifs. Ils définissent la statique de l'interface.

Le schéma de la conversation permet de décrire la dynamique du dialogue entre l'application interactive et l'utilisateur. Elle est nécessaire si des contraintes doivent être posées sur les opérations que l'utilisateur pourra effectuer.

Le schéma de la dynamique d'implémentation constitue un bon résumé des différentes étapes qui le précèdent. Il est tout autant utile au concepteur qui disposera de la sorte de

[1] Mais s'il faut ajouter de nouvelles interfaces ?! La solution est de fournir une copie de toutes les données de l'application. Cfr section précédente.

l'enchaînement que devra respecter son implémentation, qu'à l'utilisateur désireux de connaître les mécanismes qui régissent l'application.

VI.6. Conclusion

Le test de notre méthodologie nous a permis de préciser, valider ou modifier les éléments de spécifications que nous avions définis au chapitre IV[1]. Les différentes phases qui constituent la méthodologie permettent de concevoir progressivement l'interface de l'application interactive.

La première étape est de traduire les informations sémantiques contenues dans les messages fonctionnels en unités de dialogue pour l'utilisateur. Aux messages interactifs fonctionnels résultants, on associera les opérations nécessaires à l'utilisateur pour les manipuler. Le schéma de la conversation permettra de définir l'ordre que celui-ci devra respecter lorsqu'il emploiera ces différentes opérations.

Lorsque toutes les informations que manipulera l'interface ont été définies, la dernière étape est de spécifier la présentation qui les visualisera.

Les éléments de définition de l'interface récoltés permettent de passer à la phase de conception et d'implémentation en connaissant les objets interactifs qui seront présentés et les actions que l'utilisateur pourra effectuer. Pour accélérer le processus d'itération, nos spécifications pourraient être à la base de la création d'une version exécutable de l'interface. Combinée au prototype des fonctionnalités, nous aurions alors la possibilité de créer rapidement une application interactive respectant les desiderata de l'utilisateur et effectuant correctement ses fonctionnalités. Nous examinons cette perspective dans le chapitre concluant ce travail.

[1] Cfr supra.

Conclusion

Ce travail nous a permis d'exposer et d'évaluer une nouvelle méthodologie de spécification d'interfaces homme-machine. Elle constitue une extension de la méthodologie de spécifications fonctionnelles proposée par F. Bodart et Y. Pigneur.

Après avoir présenté les nombreux problèmes liés à la conception d'une application interactive, nous avons analysé les outils qui étaient offerts au concepteur pour faciliter et accélérer son processus de réalisation d'une application interactive. Ceci constituait le contexte de notre travail.

Nous avons ensuite posé les bases de notre méthodologie avant de définir les divers éléments qui la constituaient. Une application interactive peut se décomposer en, d'une part, ses fonctionnalités et, de l'autre, son ou ses interfaces. La spécification d'une de celles-ci débutera par l'analyse de la Tâche. Il s'agit d'une étape intermédiaire permettant de décrire la statique (les messages interactifs) et la dynamique (le schéma de la conversation) de l'interface qui sera réalisée.

La Tâche est indépendante de l'implémentation qui en sera faite. Le schéma de la conversation permet de décrire l'enchaînement des messages qui seront saisis par l'utilisateur ou affichés par l'interface.

Le schéma de la dynamique d'implémentation constitue un résumé à la fois de l'analyse fonctionnelle et de l'analyse de la Tâche. Plus précisément, il intègre le schéma de la conversation au schéma de la dynamique des traitements. Il peut constituer une aide à l'utilisateur qui désire connaître le fonctionnement de l'application.

La dernière phase de la méthodologie consiste en la Présentation des informations qui ont été définies lors de l'analyse de la Tâche. Elle se fera à l'aide d'objets interactifs sur lesquels l'utilisateur pourra exécuter des actions physiques.

Nous avons procédé à l'expérimentation de notre méthodologie en vue de tester les hypothèses que nous avions posées lors de son élaboration :

- La première de nos hypothèses était **l'indépendance des fonctionnalités par rapport à l'interface**. Cette indépendance peut se faire au niveau de l'implémentation de l'application interactive ou de sa spécification. L'indépendance de spécification n'est pas souhaitable car les informations que les deux composantes s'échangeront proviendront tout autant des besoins de l'utilisateur que de ceux des fonctions de l'application.

Par contre, l'indépendance modulaire est indispensable pour la création de plusieurs interfaces partageant les mêmes fonctionnalités. Des difficultés peuvent survenir si les interfaces ne disposent pas d'une copie de toutes les informations que manipulent les fonctions de l'application. En effet, la création d'une nouvelle interface pourra entraîner la modification de celles-ci parce que les informations dont elle a besoin ne lui sont pas toutes communiquées.

- Notre deuxième hypothèse était la possibilité de **spécifier de manière complète une interface en ayant préalablement spécifié les fonctionnalités de l'application.** Cette approche semble inadaptée à la spécification d'une application ayant parmi ses objectifs d'être adaptée à l'utilisateur. Nous avons proposé une approche mixte où les deux spécifications s'effectueraient en parallèle. Le but de cette approche est de permettre la réalisation de notre premier objectif : les messages fonctionnels qui seront définis contiendront toutes les informations que nécessitent les fonctions de l'application ou ses différentes interfaces.

- Nous avons également pris comme hypothèse que la **spécification de la Tâche pouvait se faire indépendamment de toute implémentation future.** Nous avons montré que cette indépendance pouvait s'effectuer sans faire de choix quant au style d'interaction ou aux types de synchronisation qui seraient utilisés. Ces choix seront effectués lors de la spécification de la Présentation et rendront inutiles les messages interactifs correspondant aux alternatives qui n'ont pas été retenues.

- Notre dernière hypothèse était avant tout un souhait : les **spécifications résultant de notre méthodologie doivent être complètes et simples à concevoir.** L'expérimentation que nous avons suivie nous permet d'estimer qu'elles réalisent ces deux objectifs. L'analyse de la Tâche constitue en quelque sorte la définition abstraite de l'interface. Elle permet de concevoir la statique de l'interface, à l'aide des messages interactifs et des opérations qu'on peut leur associer, et sa dynamique, grâce au schéma de la conversation. La spécification de la Présentation est la concrétisation de la Tâche.

  Plutôt que d'être incomplète, il nous a semblé que la spécification de la Présentation était quelque peu fastidieuse. Le fait de définir "sur papier" les caractéristiques de tous les objets interactifs et les actions par lesquelles l'utilisateur pourra les manipuler n'est pas désirable si l'on veut atteindre rapidement une version exécutable de l'application interactive. Un moyen de l'éviter serait d'utiliser un outil tel que Hypercard pour concrétiser visuellement l'analyse de la Tâche Nous proposons donc que **la phase de spécification de la Présentation soit remplacée par une phase de spécification interactive des objets et des actions associées pour obtenir directement un prototype de l'interface.**

L'évaluation des différentes hyptohèses qui étaient à la base de notre méthodologie nous permet d'énoncer les modifications qui permettront de l'améliorer. Elles tiennent principalement en deux points :

Le premier concerne la définition des informations que se communiqueront les deux composantes d'une application interactive c'est-à-dire les messages fonctionnels. Ceux-ci devront correspondre à l'ensemble des informations que manipulent l'application proprement dite. Il sera dès lors possible de répondre à toutes les questions que peut se poser l'utilisateur, qu'elles concernent le contexte de l'application ou l'aide qu'il désire recevoir. Cet objectif nécessite la prise en compte des besoins de l'utilisateur dès la phase initiale de conception de l'application.

Le second tient à la rapidité de conception de l'interface pour faciliter et accélérer le processus d'itération. La phase de spécification de la Présentation ralentissait fortement cet objectif. Elle peut être remplacée par une définition interactive de ses éléments constitutifs. L'analyse de la Tâche, qui représente la définition abstraite de l'interface, permet d'y parvenir.

Pour conclure ce travail, nous voulons à nouveau souligner la place primordiale qu'occupe dorénavant l'utilisateur dans la conception d'une application interactive : il est le juge final du produit, celui dont l'avis décidera de l'avenir du logiciel. Ce dernier devra l'aider dans sa tâche et non lui dicter son comportement, lui être adapté, être d'un apprentissage aisé, être convivial... L'objectif est de taille mais le résultat en vaut la chandelle!

Annexe 1 :
Présentation de l'exemple de saisie d'une commande

1.0. Introduction

L'exemple que nous avons utilisé pour expérimenter notre méthodologie consiste en la phase d'enregistrement d'une commande. Nous n'avons pas posé d'hypothèse sur le type de produit ou le type d'entreprise pour lesquels il serait implémenté. Notre but était de disposer d'un cadre d'expérimentation susceptible de nous fournir des éléments d'évaluation de la méthodologie sans pour autant réaliser une application interactive efficace.

Les spécifications fonctionnelles sont présentées au chapitre V. Nous les présentons ici de manière informelle.

1.1. Spécifications informelles de l'exemple

L'enregistrement d'une commande client se décompose en plusieurs étapes :

- l'opérateur peut fournir soit le nom, soit le numéro du client. L'introduction du nom du client signifie qu'il ne possède pas encore de numéro. Il s'agit dès lors d'un nouveau client. Une fonction de mémorisation permettra de lui attribuer un numéro. Le numéro d'un client permet de l'identifier parmi l'ensemble des personnes achetant des produits à l'entreprise et constitue l'information qui devra être saisie lorsqu'il passe une commande. Il y aura une vérification de ce numéro pour qu'il corresponde effectivement à une personne enregistrée dans la base de données de la firme et on vérifiera également qu'il n'est pas sur la liste noire (les mauvais payeurs).

- l'opérateur devra spécifier l'adresse d'un nouveau client pour que l'entreprise sache où acheminer les marchandises et les prospectus qu'elle pourrait envoyer. Il pourra également modifier l'adresse d'un ancien client qui a déménagé. Une procédure d'enregistrement de l'adresse permettra d'y parvenir

- les lignes de la commande seront constituées d'un numéro de produit et de la quantité désirée. Chaque numéro de produit devra correspondre à un produit vendu par la firme et la quantité commandée ne devra pas dépasser la quantité disponible (à moins qu'il n'existe un produit de substitution disponible dans les quantités adéquates).

- si les références du client ont été introduites avec succès et qu'au moins une ligne de commande a pu être validée, la commande sera alors enregistrée, à moins que le montant de la commande ne dépasse la limite d'achat du client.

1.2. Implémentation de l'exemple

La phase d'enregistrement que nous avons réalisée dispose de deux interfaces correspondant à deux types de saisie différents :

- l'opérateur pourra enregistrer une commande écrite, envoyée par le client. Ce dernier aura rempli le bon de commande, accompagnant par exemple le catalogue qu'il a reçu, et l'aura communiqué à la firme.

- la firme disposera de standardistes permettant aux clients de passer des commandes par téléphone. Le client sera alors directement en contact avec l'opérateur et spécifiera interactivement la commande qu'il souhaite.

Ces deux types d'interfaces sont différents à plusieurs égards. Le chapitre V analyse leurs caractéristiques et leurs implications.

Annexe 2 :
Spécifications pour Turbo/Pascal

2.0. Introduction

Turbo/Pascal est un environnement de programmation sur compatibles IBM[1]. Il permet l'édition, la compilation et l'exécution d'applications écrites en Pascal. Le style d'interaction qui pourra être offert par l'application réalisée sera généralement le langage de commande.

Turbo n'est pas un outil d'aide à la conception d'application interactive[2]. Nous l'avons choisi pour tester nos spécifications sur un environnement ne permettant pas la manipulation directe. De la sorte, il ne nous offre aucun objet interactif standard sur lesquels nous pourrions nous baser. Toute la Présentation est à la charge du concepteur. Puisqu'il n'existe pas d'objet standard, l'élément de spécification qui définissait le type de l'objet devient inutile.

Nous nous sommes basés sur le bon de commande de la firme (voir figure 1)pour définir la Présentation des messages interactifs (voir figure 1). La visualisation des informations constituera nos objets interactifs.

NUMERO :		
NOM :	PRENOM :	TITRE :
Rue :		Numéro :
Code postal :		Localité :
Lignes de commande :		
Numéro de produit :	Quantité :	Libellé :
Numéro de produit :	Quantité :	Libellé :
Numéro de produit :	Quantité :	Libellé : .....
TOTAL:		Rabais :

Figure 1 : le bon de commande qui sera présenté à l'écran

Voici la spécification de nos objets interactifs :

[1] Il existe également pour d'autres environnements mais nous ne nous y intéresserons pas.

[2] Hypercard n'a pas non plus été conçu dans cet objectif mais le langage qu'il offre permet d'y parvenir aisément.

2.1. Spécification des objets interactifs

Num_client

Nom : Num_client;
Contenu ou visualisation : un rectangle d'une ligne avec à l'intérieur le texte "Numéro";
Justification : l'objet correspond à la visualisation de l'information sur le bon de commande;
Message interactif qu'il représente : Num_client_à_saisir et Num_client_à_afficher;
Opérations portant sur le contenu sémantique :

- sélectionner le message --> choisir l'alternative correspondant au message lors de l'apparition du message choix_saisie;
- affecter une valeur à l'attribut du message--> introduire une valeur à l'aide du clavier;
- corriger la valeur de l'attribut --> utiliser la touche <delete> du clavier et introduire les modifications;
- effacement de la valeur de l'attribut --> utiliser la touche <delete> le nombre de fois suffisant pour effacer la valeur;
- clôturer le message : taper sur la touche <return>;

Contraintes d'enchaînement : l'utilisateur ne peut affecter une valeur à l'objet que si l'objet Nom_client ne contient pas de valeur;
Fonctions déclenchées : validation_client et validation_provenance;

Nom_client

Nom : Nom_client;
Contenu ou visualisation : un rectangle d'une ligne. La ligne est divisée en trois partie : l'une pour le nom, une autre pour le prénom et la troisième pour le titre (avec les textes : "Nom", "Prénom", "Titre");
Justification : l'objet correspond à la visualisation des informations sur le bon de commande;
Message interactif qu'il représente : les messages Nom_client_à_saisir et Nom_client_à_afficher;
Opérations portant sur le contenu sémantique :

- sélectionner le message --> choisir l'alternative correspondant au message lors de l'apparition du message choix_saisie;
- affecter une valeur à un attribut du message--> introduire une valeur à l'aide du clavier;
- corriger la valeur d'un attribut --> utiliser la touche <delete> du clavier et introduire les modifications;
- effacement de la valeur d'un attribut --> utiliser la touche <delete> le nombre de fois suffisant pour effacer la valeur;
- clôturer le message : taper sur la touche <return> après la saisie de chaque attribut (la terminaison de la saisie du dernier déclenchera la clôture du message);

Contraintes d'enchaînement : l'utilisateur ne peut affecter une valeur à l'objet que si l'objet Num_client ne contient pas de valeur;
Fonctions déclenchées : mémorisation_client;

Adr_domicile

Nom : Adr_domicile;
Contenu ou visualisation : un rectangle de deux lignes. La première contient la rue et le numéro, la deuxième le code postal et la localité (avec les textes correspondants);
Justification : l'objet correspond à la visualisation des informations sur le bon de commande;
Messages interactifs qu'il représente : message adr_domicile_à_saisir et adr_domicle_à_aficher;
Opérations portant sur le contenu sémantique :

- sélectionner le message --> choisir l'alternative correspondant au message lors de l'apparition du message modif_adresse si le numéro a été préalablement saisi (la sélection sera automatique si le nom a été saisi);
- affecter une valeur à un attribut du message--> introduire une valeur à l'aide du clavier;
- corriger la valeur d'un attribut --> utiliser la touche <delete> du clavier et introduire les modifications;
- effacement de la valeur d'un attribut --> utiliser la touche <delete> le nombre de fois suffisant pour effacer la valeur;
- clôturer le message : taper sur la touche <return> après la saisie de chaque attribut (la terminaison de la saisie du dernier déclenchera la clôture du message);

Contraintes d'enchaînement : l'adresse du client ne peut être saisie que si le numéro ou le nom du client a été préalablement saisi. Sa saisie est obligatoire si le nom du client a été saisi;
Fonctions déclenchées : modification_adresse;

Lignes_de_commande

Nom : Lignes_de_commande;
Contenu ou visualisation : un rectangle avec une ligne pour le titre (le texte "Lignes de commande") et un nombre de lignes supplémentaires composées des texte "numéro de produit :", "quantité :" et "libellé :";
Justification : l'objet correspond à la visualisation des informations sur le bon de commande;
Message interactif qu'il représente : les messages Num_produit, Qu_produit et Libellé_produit;
Opérations portant sur le contenu sémantique :

- sélectionner un message --> l'utilisateur ne peut sélectionner un message car la gestion est réalisée par l'interface (elle positionne le curseur à l'attribut numéro de produit);
- affecter une valeur à un attribut du message--> introduire une valeur à l'aide du clavier;
- corriger la valeur d'un attribut --> utiliser la touche <delete> du clavier et introduire les modifications;
- effacement de la valeur d'un attribut --> utiliser la touche <delete> le nombre de fois suffisant pour effacer la valeur;
- clôturer un message : taper sur la touche <return> après la saisie de chaque attribut (la terminaison de la saisie du dernier déclenchera la clôture du message);

Contraintes d'enchaînement : l'objet ne pourra prendre de valeur qu'après la saisie des références du client;
Fonctions déclenchées : validation_produit et validation_ligne;

Choix_saisie

Nom : Choix_saisie;
Contenu ou visualisation : un menu qui apparaîtra avec les deux options "Nouveau client", "Ancien client") et le texte "Taper N ou A".
Justification : l'objet représente la possibilité qui est offerte à l'utilisateur de saisir les références d'un nouveau ou d'un ancien client;
Message interactif qu'il représente : choix_saisie;
Opérations portant sur le contenu sémantique :

- sélectionner un attribut du message --> taper N ou A selon que l'utilisateur est Nouveau ou Ancien;
- clôturer le message --> le message est automatiquement clôturé après la sélection d'un attribut;

Contraintes d'enchaînement : l'objet apparaîtra au début de l'application;
Fonctions déclenchées : aucune;

Modif_adresse

Nom : Modif_adresse;
Contenu ou visualisation : un message qui apparaîtra avec la question "Faut-il modifier l'adresse (taper Oui ou Non)?";
Justification : l'objet représente la possibilité qui est offerte à l'utilisateur de modifier l'adresse d'un ancien client;
Message interactif qu'il représente : AUCUN;
Opérations portant sur le contenu sémantique :

- sélectionner un attribut du message --> taper O ou N selon qu'il faut ou non modifier l'adresse;
- clôturer le message --> le message est automatiquement clôturé après la sélection d'un attribut;

Contraintes d'enchaînement : l'objet apparaîtra lorsque l'utilisateur a terminé la saisie du nom du client;
Fonctions déclenchées : aucune;

Message_aide

Nom : Message_aide;
Contenu ou visualisation : un texte correspondant au contenu du message (le texte sera présenté dans l'entièreté de l'écran);
Justification : l'écran permet bien sûr d'afficher un nombre important d'information en même temps;
Message interactif qu'il représente : Message_aide;
Opérations portant sur le contenu sémantique :

- ranger le message --> taper sur n'importe quelle touche;
- sélectionner le message --> taper <contrôle> ?;

Contraintes d'enchaînement : aucune;
Fonctions déclenchées : aucune;

Montant_total

Nom : Montant_total;
Contenu ou visualisation : un rectangle d'une demi-ligne avec le texte "Total :";
Justification : l'objet correspond à la visualisation de l'information sur le bon de commande;
Message interactif qu'il représente : message total;
Opérations portant sur le contenu sémantique :
- il ne peut être manipulé par l'utilisateur;

Contraintes d'enchaînement : l'objet prendra une valeur lorsque la commande aura été validée;
Fonctions déclenchées : aucune;

Rabais

Nom : Rabais;
Contenu ou visualisation : un rectangle d'une demi-ligne avec le texte "Rabais :"(sur la même ligne que l'objet Montant_total;
Justification : l'objet correspond à la visualisation de l'information sur le bon de commande;
Message interactif qu'il représente : message rabais;
Opérations portant sur le contenu sémantique :
- il ne peut être manipulé par l'utilisateur;

Contraintes d'enchaînement : l'objet prendra une valeur lorsque la commande aura été validée et si le client est membre du personnel;
Fonctions déclenchées : aucune;

Client_invalide

Nom : Client_invalide;
Contenu ou visualisation : un texte correspondant au contenu du message qu'il représente et qui apparaîtra en bas de l'écran;
Justification : tout message d'erreur sera affiché à la même place c'est-à-dire en bas de l'écran;
Message interactif qu'il représente : client_non_valide;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur n'importe quelle touche;

Contraintes d'enchaînement : l'objet apparaîtra lorsque l'erreur correspondante a été commise;
Fonctions déclenchées : aucune;

Provenance_invalide

Nom : Provenance_invalide;
Contenu ou visualisation : un texte correspondant au contenu du message qu'il représente et qui apparaîtra en bas de l'écran;
Justification : tout message d'erreur sera affiché à la même place c'est-à-dire en bas de l'écran;
Message interactif qu'il représente : le message Provenance_non_valide;

Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur n'importe quelle touche;

Contraintes d'enchaînement : l'objet apparaîtra lorsque l'erreur correspondante a été commise;
Fonctions déclenchées : aucune;

Produit_invalide

Nom : Produit_invalide;
Contenu ou visualisation : un texte correspondant au contenu du message qu'il représente et qui apparaîtra en bas de l'écran;
Justification : tout message d'erreur sera affiché à la même place c'est-à-dire en bas de l'écran;
Message interactif qu'il représente : le message Produit_non_valide;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur n'importe quelle touche;

Contraintes d'enchaînement : l'objet apparaîtra lorsque l'erreur correspondante a été commise;
Fonctions déclenchées : aucune;

Ligne_invalide

Nom : Ligne_invalide;
Contenu ou visualisation : un texte correspondant au contenu du message qu'il représente et qui apparaîtra en bas de l'écran;
Justification : tout message d'erreur sera affiché à la même place c'est-à-dire en bas de l'écran;
Message interactif qu'il représente : le message Ligne_non_valide;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur n'importe quelle touche;

Contraintes d'enchaînement : l'objet apparaîtra lorsque l'erreur correspondante a été commise;
Fonctions déclenchées : aucune;

Commande_invalide

Nom : Commande_invalide;
Contenu ou visualisation : un texte correspondant au contenu du message qu'il représente et qui apparaîtra en bas de l'écran;
Justification : tout message d'erreur sera affiché à la même place c'est-à-dire en bas de l'écran;
Message interactif qu'il représente : le message Commande_non_valide;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur n'importe quelle touche;

Contraintes d'enchaînement : l'objet apparaîtra lorsque l'erreur correspondante a été commise;
Fonctions déclenchées : aucune;

Produit_invalide

Nom : Produit_invalide;
Contenu ou visualisation : un texte correspondant au contenu du message qu'il représente et qui apparaîtra en bas de l'écran;
Justification : tout message d'erreur sera affiché à la même place c'est-à-dire en bas de l'écran;
Message interactif qu'il représente : le message Produit_non_valide;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur n'importe quelle touche;

Contraintes d'enchaînement : l'objet apparaîtra lorsque l'erreur correspondante a été commise;
Fonctions déclenchées : aucune;

Num_produit_incorrect

Nom : Num_produit_incorrect;
Contenu ou visualisation : un texte correspondant au contenu du message qu'il représente et qui apparaîtra en bas de l'écran;
Justification : tout message d'erreur sera affiché à la même place c'est-à-dire en bas de l'écran;
Message interactif qu'il représente : le message Num_produit_incorrect;
Opérations portant sur le contenu sémantique :
- supprimer le message --> cliquer sur n'importe quelle touche;

Contraintes d'enchaînement : l'objet apparaîtra lorsque l'erreur correspondante a été commise;
Fonctions déclenchées : aucune;

Commande_saisie

Nom : Commande_saisie;
Contenu ou visualisation : Voir figure 1;
Justification : l'objet est la visualisation du bon de commande à l'écran;
Message interactif qu'il représente : Commande_enregistrée;
Opérations portant sur le contenu sémantique :
- ranger le message --> taper sur n'importe quelle touche;
- sélectionner le message --> le message ne sera plus accessible via l'application une fois qu'il aura été rangé;

Contraintes d'enchaînement : le message apparaîtra lorsque la commande a été validée;
Fonctions déclenchées : aucune;

Annexe 3 :

Implémentation de la seconde interface

dans l'environnement Hypercard

La réalisation d'une application interactive à l'aide du logiciel Hypercard se fait en attachant aux objets interactifs des procédures chargées de réaliser les fonctions de l'application (cfr section VI.4.1.0.).

1. Indépendance modulaire entre les fonctionnalités et l'interface

Il eût été possible d'assurer l'indépendance modulaire entre le dialogue et l'application proprement dite grâce à la possibilité qu'offerte Hypercard d'appeler des applications en dehors de celui-ci. Il nous semblait cependant plus aisé de concevoir les fonctions de l'application à l'intérieur d'hypercard. L'indépendance modulaire pouvait de même être réalisée à l'intérieur d'Hypercard en regroupant l'ensemble des fonctions de l'application au niveau du background d'une carte du stack commande_à_saisir[1]. Ceci était rendu plus difficile par le fait que le concept de passage de paramètres n'existe pas dans Hypercard. Il nous a finalement paru préférable de garder la structure naturellement répartie de l'application en associant aux objets interactifs les fonctions qu'ils déclenchent (voir la spécification des objets interactifs). Ceci aurait pu posé des problèmes si plusieurs objets appelaient la même fonction[2].

Nous n'avons donc pas d'indépendance modulaire au sens strict mais nous verrons que chaque fonction est bien délimitée à l'intérieur des scripts (procédures) associés aux objets interactifs.

2. Méthode suivie pour l'implémentation à partir des spécifications

Le résultat des spécifications nous a donné d'une part l'analyse fonctionnelle nécessaire à la réalisation des fonctions de l'application et, d'autre part, la spécification des objets interactifs représentant la tâche à l'utilisateur.

Nous avons vu qu'Hypercard utilise une approche orientée objet ce qui nous permet d'implémenter l'application interactive par construction successive des objets interactifs.

2.1. Construction de la Base de Données

Dans un premier temps, nous avons construit la base de données correspondant au schéma Entité-Association de l'analyse fonctionnelle. Il faut remarquer qu'Hypercard n'est pas un SGBD (Système de Gestion de Base de Données) et que la BD a du être réalisée de manière manuelle : un stack représente un type d'entité, chaque carte représentant l'occurence d'une entité avec des champs représentant les attributs et les associations étant représentées par des boutons permettant de relier une occurence d'entité à une autre. Une implémentation plus performante de la BD pourrait être réalisée grâce à un véritable SGBD qui serait appelé d'Hypercard grâce à son ouverture à d'autres applications.

---

[1] Chaque carte possède alors le même code, correspondant aux fonctionnalités

[2] Il y aurait alors eu une répétition de la fonction à l'intérieur de chaque objet qui la déclenche.

En résumé, nous avons défini les stacks :

- Clients
- Commandes
- Produits

Une carte **Client** est constituée

des champs :

- Num_client
- Nom_client
- Adr_client
- Lim_achat
- Catégorie

et des boutons :

- Home
- Commande : pour passer à la dernière commande du client
- Produit : pour passer au stack Produits
- Client précédent
- Client suivant

Une carte **Produit** est constituée

des champs :

- Num_produit
- Libellé
- Unité-achat
- Prix_unitaire
- Etat_appro
- Date_réappro

et des boutons :

- Commande : donne la liste des commandes portant sur le produit
- Clients : pour passer au stack Clients
- Substitution : donne pour un produit 'épuisé' un de ses produits analogues
- Home
- Produit suivant
- Produit précédent

Une carte **Commande** est constituée

des champs :

- Num_commande
- Num_client
- Lignes_commande
- Montant_total
- Rabais

et des boutons :

- Home
- Client : pour passer au client ayant passer la commande
- Produits : pour passer au produit spécifié
- Commande précédente
- Commande suivante

Ceci constitue notre B.D., construite à partir du schéma Entité-Association de l'analyse fonctionnelle. Rappelons que pour une réelle efficacité l'usage d'un S.G.B.D. se révèle indispensable.

2.2. Construction de l'application interactive

La construction de l'application interactive se base sur la spécification des objets interactifs de notre seconde interface (cfr section VI.4.1.3.).

2.2.1. Création des objets

Un premier lecture de tous les objets interactifs de l'application nous a permis de les dessiner à l'écran. Le résultat de cette étape est la création du stack Commande_à_saisir, du stack Erreurs et du stack Aide.

Une carte du stack **Erreurs** représente un objet interactif dont le message interactif correpondant est de type erreur sémantique[1] Nous avons ainsi les cartes :

- Client_invalide
- Provenance_invalide
- Produit_invalide
- Ligne_invalide
- Commande_invalide

Ces cartes sont affichées quand l'erreur correspondante est survenue et le fait de cliquer sur la carte ramène l'utilisateur à l'application.

[1] Rappelons qu'un message d'erreur syntaxique sera créé lors de l'occurence de l'erreur et ne sera donc pas enregistré dans un stack.

Une carte commande_à_saisir est la visualisation de l'application interactive. Grâce à l'approche orientée objet, il est tout-à-fait possible d'effectuer plusieurs saisies de commande en même temps en travaillant sur plusieurs cartes commande_à_saisir. Cette forme de parallélisme est un élément supplémentaire de la qualité de l'interface qui sera proposée.

Une carte **Commande_à_saisir** est constituée

des champs :

- Num_client
- Nom_client
- Adr_domicile
- Lignes_de_commande
- Montant_total
- Rabais

et des boutons :

- OK
- Annuler
- Home
- Aide_saisie
- Commande_avant
- Commande_après

Chaque champ ou bouton représente un objet interactif. Les boutons Home, Commande_avant et Commande_après[1] n'ont pas été spécifiés lors de la Présentation car ils constituent des possibilités intrinsèques à Hypercard, quelque soit l'application réalisée.

L'objet interactif Message_aide sera représenté par une carte dans le stack Aide (qui pourra ultérieurement s'enrichir de nouveaux messages).

Le résultat du bon déroulement de l'application interactive c'est-à-dire la validation de la commande produira une nouvelle carte du stack Commande.

2.2.2. Implémentation des objets

Après avoir dessiné, et ainsi créé les objets, il faut à présent leur attacher le code qui réalisera les fonctionnalités du dialogue. Chaque objet interactif se verra associer un script qui réalisera ses fonctionnalités.

[1] Home permet de retourner au menu principal d'Hypercard, Cmde_avant et Cmde_après déplace l'opérateur d'une carte en avant ou en arrière dans le stack Commande_à_saisir.

Reprenons chaque objet interactif avec ce qu'il devra réaliser :

- **Num_client** :

Lorsque l'utilisateur a entré une valeur pour ce champ, il faut :
- vérifier que Num_client est un entier
- afficher le message d'erreur s'il ne l'est pas
- déclencher la fonction de validation-client
- afficher le message d'erreur si le client est invalide
- afficher Nom_client et Adr_domicile si le client est valide
- déclencher la fonction de validation-provenance
- afficher le message d'erreur si la provenance est invalide
- interdire la saisie de Nom_client

Rappelons que nous avons décidé d'implémenter les fonctionnalités de l'application dans le code de nos objets interactifs tout en gardant une séparation entre le code du dialogue et le code des fonctions.

Les informations nécessaires pour connaître ce qui devait être fait par cet objet interactif proviennent de la spécification de l'objet ainsi que de la dynamique d'implémentation.

- **Nom_client** :

Lorsque l'utilisateur a entré une valeur pour ce champ, il faut :
- interdire la saisie de Num_client
- activer la fonction mémorisation-client
- afficher le nouveau Num_client renvoyé par mémorisation-client

- **Adr_domicile** :

Lorsque l'utilisateur a entré une valeur pour ce champ, il faut :
- vérifier que le NPA est un entier
- afficher le message d'erreur s'il ne l'est pas
- activer la fonction modification-adresse

- **Lignes_de_commande** :

Pour chaque ligne que l'utilisateur a saisi, il faut :
- vérifier que Num_produit est correct
- afficher le message d'erreur s'il ne l'est pas
- activer la fonction validation-produit
- afficher le message d'erreur si le produit est invalide
- vérifier que la quantité est entière positive
- afficher le message d'erreur si elle ne l'est pas
- afficher le libellé correspondant au numéro de produit

- **Montant_total** :

- interdire la saisie par l'utilisateur

- **Rabais** :
  - interdire la saisie par l'utilisateur

- **OK** :

Lorsque l'utilisateur a cliqué sur le bouton, il faut :
- vérifier qu'il y a au moins une ligne de commande valide
- afficher le message d'erreur si ce n'est pas le cas
- vérifier que soit le nom, soit le numéro du client a été saisi
- afficher le message d'erreur si ce n'est pas le cas
- activer la fonction validation-commande
- afficher le message d'erreur si la commande est invalide
- afficher le montant total
- afficher le rabais s'il existe

- **Message_aide** :

Lorsque l'utiliateur a cliqué sur le bouton :
- afficher le message d'aide

3. Evaluation de la méthodologie avec l'environnement Hypercard

Le passage des spécifications à l'étape de conception et d'implémentation n'a posé aucun problème majeur. Ce qui nous intéresse est, bien sûr, les difficultés qui ont pu survenir au niveau de la conception de l'interface. Une première conclusion tient à la conception de haut niveau de dialogue réalisable à l'aide d'Hypercard. Une grande partie du code à réaliser pour implémenter un dialogue tient à l'analyse des évènements utilisateur. Une fois l'évènement analysé, le travail consiste à appeler les fonctions de l'application qui y correspondent si l'évènement a une incidence fonctionnelle ; sinon, le dialogue se charge entièrement de l'évènement.

La puissance d'Hypercard tient à la facilité de gestion des évènements utilisateur. La plupart du code du dialogue se trouve dans les couches inférieures d'hypercard et ne doit donc pas être programmé. Par exemple, un click souris sur un objet sera directement acheminé vers l'objet correspondant qui exécutera les actions spécifiées lors de l'occurence de l'évènement de type click souris. Par contre, dans une application dans un langage standard tel que Pascal ou C, l'évènement devra être analysé pour connaître sur quel objet il a porté de même qu'il faudra analyser la nature de l'évènement.

Un autre exemple est de considérer la facilité de réalisation d'un objet interactif. Avec Hypercard, il suffit de créer un objet du type désiré (par l'utilisation d'un menu) et d'ensuite lui donner la forme ou les caractéristiques désirées à l'aide d'un mécanisme de manipulation ou d'une table reprenant les caractéristiques de l'objet. Il n'y a donc aucune programmation; cela se fait interactivement.

En résumé, le haut niveau de conception d'Hypercard nous a permis de créer rapidement un dialogue nécessitant un minimum de codage (typiquement, les actions de gestion d'objets

tels que la navigation ou l'affichage) tandis que la plus grande partie du code sert à la réalisation de l'application proprement dite (les fonctions).

Notons cependant que la puissance d'Hypercard peut devenir gênante lorsqu'on veut restreindre les possibilités de l'utilisateur (respecter le schéma de la conversation) car la réduction de la liberté d'action de l'utilisateur entraîne un supplément de code pour la réaliser. Ceci peut avoir pour conséquence un oubli de la part du concepteur de certaines restrictions à effectuer. En bref, au lieu de décrire le schéma de la conversation c'est-à-dire le passage d'une opération à une autre, on décrit ce qui est interdit après l'accomplissement d'une opération.

Un autre point qui mérite d'être analysé est l'incidence des spécifications de l'interface sur la conception. Nous sommes tout-à-fait positifs sur ce point. La spécification du dialogue nous a permis d'obtenir et de clarifier toutes les informations nécessaires à la création du dialogue. Une petite remarque cependant : la connaissance du logiciel sur lequel sera implémentée l'application est nécessaire pour spécifier complètement l'interface. L'usage d'un logiciel de haut niveau peut permettre l'extension du dialogue à des fonctionnalités supplémentaires telles que, pour Hypercard, le passage à la commande suivante ou précédente. La connaissance du logiciel peut ainsi avoir des conséquences sur la qualité du dialogue et sur sa spécification.

4. Conclusion sur l'implémentation avec l'environnement Hypercard

Les spécifications de l'interface et des fonctionnalités nous ont donné une base solide pour la conception et l'implémentation de l'application interactive. Une connaissance préalable de l'environnement utilisé nous a permis d'étendre l'interface pour offrir à l'utilisateur de plus grandes possibilités[1].

5. Evaluation de l'outil de conception

Nous avons examiné au chapitre II une taxonomie d'évaluation des SGD. Hypercard n'est pas à proprement parler un SGD car il ne propose pas un langage de spécification à partir duquel il implémenterait l'application interactive. Cependant, le langage de haut niveau qu'il offre (Hypertalk est un langage semi-naturel) permet de réaliser le code de l'interface assez rapidement.

5.1. Niveau d'abstraction

Nous avons signalé que le code nécessaire à implémentation de l'interface était relativement limité grâce à la gestion qu'effectue Hypercard des événements-utilisateur et leur communication aux objets concernés. L'interface qui sera conçue aura peu de transformations à faire en vue de communiquer aux fonctionnalités des informations d'un haut niveau d'abstraction.

5.2. Localisation du contrôle

Le contrôle est mixte : l'utilisateur dirige l'application mais les fonctionnalités prennent le contrôle dès qu'une action de l'utilisateur a déclenché le traitement d'une fonction. Celle-ci

[1] Passer à la commande précédente ou suivante et ainsi travailler en parallèle.

peut alors entreprendre les actions qui auront été définies et qui pourront comprendre l'envoi à l'utilisateur d'une demande de renseignements supplémentaires...

5.3. Ordonnancement des événements

L'ordonnancement est implicite[1] : on spécifie le comportement de chaque objet sans définir explicitement les enchaînements qui auront lieu. L'approche orientée objet fait en sorte qu'il n'existe pas de coordinateur ou autre mécanisme d'ordonnancement; chaque objet est programmé pour répondre aux actions que l'utilisateur effectuera sur lui, sans se soucier des autres objets de l'application.

5.4. Adaptabilité

Voici sans doute l'atout majeur d'Hypercard. La présentation de l'interface est modifiable interactivement par l'utilisateur. Il peut le faire lors du déroulement de l'application ou en dehors de celle-ci. Pour cela, il passe du mode utilisateur au mode champ. Il peut alors modifier chaque objet selon ses désirs[2]. Il peut même modifier le code de l'interface en passant au mode script. Il doit alors bien sûr posséder des connaissances dans le langage Hypertalk.

5.5. Parallélisme

Les possibilités de parallélisme sont assez étendues puisqu'il est par exemple possible pour notre exemple de travailler sur plusieurs commandes en même temps, de communiquer avec d'autres utilisateurs moyennant un modem et bien sûr de manipuler en parallèle les différents objets de l'application.

5.6. Généralité

Hypercard permet de créer toute application interactive réalisable sur Macintosh. Elle doit cependant être conçues selon l'approche orientée objet puisqu'Hypercard ne connaît pas la notion de programme. Il est certain que c'est loin d'être un désavantage pour la conception d'une interface.

5.7. Contexte

Pour rester en accord avec l'état-de-l'art, Hypercard n'offre aucune gestion du contexte de l'application. Rappelons cependant qu'il n'est pas un SGD[3] et n'a donc pas de motif pour permettre une telle gestion. Le concepteur aura la charge de la réalisation de cet élément.

---

[1] Voir même inexistant, selon l'option que choisit le concepteur.

[2] Nous avions dit que la conception de l'application se faisait interactivement. Le mode champ et le mode script, dont nous parlons ci-dessus, sont en réalité le moyen de réaliser l'application. L'utilisateur devient alors le concepteur du système.

[3] Bien que son évaluation le placerait à une bonne position dans le hit-parade des SGD de haut niveau.

5.8. Conclusion

Hypercard se révèle étonnament efficace par rapport à la taxonomie que nous avons énoncée. Son adaptabilité, son parallélisme et son haut niveau d'abstraction permettent la création d'applications interactives réalisant les objectifs que nous avions définis aux deux premiers chapitres. Un tableau récapitulatif des différents outils dont nous avons parlé est présenté ci-dessous.

	MacApp	Use	Omega	Peridot	Hypercard
Abstraction	défini par concepteur	défini par concepteur	haut niveau	haut niveau	haut niveau
Contrôle	externe ou mixte	externe	mixte	mixte	mixte
Ordonnanc.	----------	explicite	explicite	implicite	implicite
Adaptabilité	semi-adaptable	à charge du concepteur	semi-adaptable	oui	oui
Parallélisme	entre objets	non	possible	possible	oui
Généralité	élevé	interrogation de BD	interrogation de BD	manipulation directe	manipulation directe
Contexte	non	non	faible	non	non

Figure 3.1. : Tableau récapitulatif des outils de conception

Bibliographie

- Ahlsen, Wilfried J. (1971). **User Engineering Principles for Interactive Systems**
Proceedings of the Fall Joint Computer Conference,
39AFIPS Press, Montvale, NJ, pp 523-532

- Baecker, R.M. & Buxton W.A.S. **Readings in Human-Computer Interaction : A Multidisciplinary Approach,**
Morgan Kaufmann Publishers, Inc.95 First Street, Los Altos, California 94022

- Bergerol, Claude. **Initiation à l'informatique**
Collection Cadrèco. Entreprise Moderne d'édition, Paris 1971

- Bernard, Sophie (1987) Projet Mouse, **Outils pour la construction d'interfaces homme-machine**
Manuel d'utilisation du gestionnaire de dialogue, Septembre 87

- Bodart, François et Pigneur, Yves (1983) **Conception assistée des applications informatiques**
1. Etude d'opportunité et analyse conceptuelle
Masson 83

- Bösser, Tom. **Learning in Man-Computer Interaction : A Review of the Literature**
Research Reports : ESPRIT. Project 385-HUFIT-Vol 11987 Springer-Verlag

- Buxton, W. & Shniderman, R. (1980). **Iteration and the Design of the Human-Computer Interface**
Proc. of the 13th Annual Meeting of the Human Factors Association of Canada, pp 72-81

- C. Rose et al. (1986). **Inside Macintosh** Addison Wesley Publ., 1986

- Card, S.K., Moran, T.P. & Newell, A. (1980a). **The Keystroke-Level Model for User Performance Time with Interactive Systems,** Communications of the ACM 23(7), pp 396-410

- Card, S.K., Moran, T.P. & Newell, A. (1980b). **Computers Text-Editing : an Information Processing Analysis of a Routine Cognitive Skill,** Cognitive Psychology 12, pp 32-74

- Card, S.K., Moran, T.P. & Newell, A. (1983). **The Psychology of Human-Computer Interaction,**
Hillskale, N.J. : Lawrence Erlbaum Associates

- Chandelon, Murielle & Warnant, Geneviève. (1987) **Modélisation d'une application interactive**
.Mémoire présenté en vue du titre de Licencié et Maître en Informatique
Septembre 87

- Chernicoff, Stephen. (1985) **Macintoch TM Revealed**
Volume one : Unlocking the toolbox, Hayden, Apple Press,
Hasbrouck Heights, New Jersey, USA, 1985

- Chritine Collet :(1987) Thèse **Les formulaires complexes dans les BD multimédia**
Laboratoire de Génie Informatique, Grenoble 87

- Coons, S.A. (1963). **An outline of the Requirements for a Computer-Aided Design System.**
AFIPS Conference Proceedings 23, 299L-304

- Coutaz , Joelle (1986a) **La construction d'interfaces homme-machine**
Rapport de recherche, Novembre 86

- Coutaz , Joelle (1986b) **The construction of User Interfaces and the object paradigm**
ECOOP1987

- Coutaz , Joelle (1987a) **La construction d'interface-utilisateur**
First European Software Engineering Conference
Strasbourg, septembre 87

- Coutaz , Joelle (1987b) **PAC, an Object Oriented Model For Dialog Design** Interact '87 Conference

- Davis, Ruth M. (1966). **Man Machine Communication**
In Cuadra, C.A. (Ed) Annual Review of Information Science and Technkology 1, Interscience, New-York, 221-254

- Diaper, Dan (1986). **Identifying the Knowledge Requirements of an Expert System's Natural Processing Interface**, in People and Computers : Designing for Usability, Proceedings of the Second Conference of the Britisch Computer Society Human Computer Interaction
Specialist Group, Britisch Computer Society Workshop Series

-. Draper (1984). **Software Engineering for User Interfaces**
in Preocessing of the 7th international Conference on Software Engineering, Orlando, FL., March 1984

- Dzida, W, Herda, S., & Itzfeldt, W.D. (1980) **A Paradigm for Task-Orientde Man-Computer Interaction.**
In R.A. Guedj, P. ten Hagen, F.R. Hopgood, H. Tucker, & D.A; Duce (Eds),
Methodology of Interaction (pp 189-193))Amsterdam : North-Holland

-. Edmonds, E.A. (1981). **Adaptative Man-Computer Interfaces,**
in Computing Skills and the User Interface, Coombs, M.J. & Alty, J.L. (eds)
Academic Press, London

- Engelbart, D.C. (1963). **A Conceptual Framework for the Augmentation of Man's Intellect.**
In Howerton & Weeks (Eds), Vistas in Information Handling,
Vol. 1, Washington, D.C. : Spartan Books, 1-29

- Engelbart, D.C. (1982). **Integrated, Evolutionary, Office Automation System**
In Landau & Bair (Eds), Emerging Office Systems, Ablex

- Engelbart, Doug (198§). **The Augmentation knowledge Workshop.**
Proceedings of the Conference on the History of Personal Workstations,
New-York : ACM, 73-83

- Fano, R.M. & Corbato, F.J. (1966). **Time-Sharing on Computers Scientific American**
214 (9), Sept. 1966, 129-140

- Faulle, Bernard. **L'informatique conversationnelle : méthodologie d'analyse et de programation.**
Les éditions d'organisation 1982

- Foley, J.D. & Van Dam, A. (1982). **Fundamentals of Interactive Computer Graphics**
Reading, Mass. : Addison-Wesley Publishing Company

- Fred Huxham, David Burnard, Jim Takatsuka (1986) **Using the Macintosh Toolbox with C**
Sybex Publ, 1986

- Gardner, H. (1987). **The Mind's New Science**
Second Edition, New York : Basic Books

- Gould, John D. & Lewis, Clayton (1985). **Designing for Usability : Key Principles and What Designers Think**
Communications of the ACM 28(3), pp 300-311

- Hayes, P.J., Szekely, P. & Lerner, R. (1985). **Design Alternatives for User Interface Management Systems Based on Experience with Cousin,** Proceedings of the CHI'85 Conference,
The Association for Computing Machinery Publ., April 1985, pp 169-175

-.Henderson, Austin Jr (1987). **The Trillium User Interface Desing Environment**, Intelligent System Laboratory, Palo Alto, in Baecker, R.M. & Buxton W.A.S. Readings in Human-Computer Interaction
A Multidisciplinary Approach, Morgan Kaufmann Publishers, Inc.95 First Street, Los Altos, California 94022

- Kernighan, Brian W. & Ritchie, Dennis M. (1984) **Le langage C**
Masson 84

- Konsynski, Kuo & Kuo, Bob (1985). **An Architecture for Dialogue Management : Implications in User-Computer Dialogue Design,** University of Arizona; Interfaces in Computing, 3, pp 259-275

- Lampson, Rutler (1986). **Personal Distributed Computing. The Alto and Ethernet Software.**
Proceedings of the Conference on the History of Personal Workstations
New-York : ACM, 101-131

- Lermigeaux, Fred (1986). **Un modèle d'architecture pour une application interactive sur Macintosh**
Rapport de stage de DESS Génie Informatique

- Licklider, J.C.R. (1968). **Man-Computer Communication**
Annual review of Information Science and Technology 3, 201-240

- Licklider, JCR (1960). **Man-Computer Symbiosis,**
IRE Transaction of Human Factors in Electronics HFE-1(1), March 1960, 4-11

- Lindsay, P.H. & Norman, D.A. (1977). **Human Information Processing : an Introduction to Psychology**
Second Edition, New York : Academic Press

- Maclean, Allan, Barnard, Phil & Wilson, Michael (1986). **Rapid Prototyping for Human Factors Research : the EASIE Approach**, in People and Computers : Designing for Usability, Proceedings of the Second Conference of the Britisch Computer Society Human Computer Interaction Specialist Group, Britisch Computer Society Workshop Series

- Mark Stefik et Daniel G. Brobow (1986). **Object-oriented Programming : Themes and Variations**
Intelligent Systems Laboratory Xerox Palo Alto Research Center
3333 Coyote Hill Road Palo Alto California 94 304

- Martin, J. (1973). **Design of Man-Computer Dialogues,**
Prentice-Hall, Inc.

- Maynard (1987) **Dec Windows : User Interface Style Guide Digital Equipment Corporation** , Massachussets, November 87

-.Metais, T. (1986). **Omega**, EDF-GDF STI/DEMA, document de présentation Omega

- Metcalfe, R.M. & Boggs, D.R. (1976). **ETHERNET : Distributed Pocket Switching for Local Computer Networks.** Communication of the ACM (1917),July 1976, 395-404

- Microsoft Windows (1985) **Software Kit. Programming Guide** Microsoft Corporation 85

- Moran, Thomas P. (1981). **The Command Language Grammar : a Representation of the User Interface of Interactive Computer Systems**, International Journal of Man-Machine Studies 15, pp 3-50

- Myers, Brad & Buxton, W. (1986). **Creating Highly-Interactive and Graphical User Interfaces by Demonstration**, in Baecker, R.M. & Buxton W.A.S. Readings in Human-Computer Interaction
A Multidisciplinary Approach, Morgan Kaufmann Publishers, Inc.95 First Street, Los Altos, California 94022

- Nelson, T.H. (1965). **A File Structure for the Complex, the Changing, and the Indeterminate**
Proceedings of the ACM National Conference, 84-100.Copyright 1965,
Association for Computing Machinery, by permission

- Nelson, T.H. (1973). **A Conceptual Framework for Man-Machine Everything**
Proceedings of the National Computer Conference, M21-M26

- Newman, W.M. & Sproull, R.F. (1979). **User Interface Design** CH28
in Principles of Interactive Computer Graphics,
Second Edition,New York : McGraw-Hill Book Company, pp 443-478

- Nievergelt, J. & Weydert, J. (1985). **Sites, Modes, and Trails : Telling the User of an Interactive System Where he is, What he can do, and How to get to places,**
in Baecker, R.M. & Buxton W.A.S. Readings in Human-Computer Interaction
A Multidisciplinary Approach, Morgan Kaufmann Publishers, Inc.95 First Street, Los Altos, California 94022

- Norman, D. (1983). **Design Principles for Human-Computer Interfaces**
Proccedings of CHI'83, pp 1-10

- Norman, D.A. (1984). **Stages and Levels in Human-Machine Interaction**, International Journal of Man-Machine Studies 21, pp 365-375

- Norman, D.A. (1985). **Four Stages of User Activities**, Human-Computer Interaction -Interact'84, pp 507-511

- Norman, D.A. (1986). **Cognitive Engineering,**
in Norman, D.A. & Draper, S.W. (1986) User Centered System Design,
Hillsdale, N.J. : Lawrence Erlbaum Associates, pp 31-61

- Petoud, Isabelle (1987).**Conception de l'interface homme-machine**
Lausanne, session de printemps 87 E.H.E.C. Université de Lausanne

- Reisner, Phyllis (1977). **Use of Psychological Experimentation as an Aid to Development of a Query Language**, IEEE Transactions on Software Engineering SE-3(3), May 1977, pp 218-229

- Reisner, Phyllis (1981). **Formal Grammar and Human Factors Design of an Interactive Graphics System,**
IEEE Transactions on Software Engineering SE-7(2), March 1981, pp 229-240

- Reisner, Phyllis (1984). **Formal Grammar as a Tool for Analysing Ease of Use : Some Fundamental Concepts,** in Thomas, J.C. & Shneider, M.L. (Eds), Ablex Publishing Corp.

- Roberts, Larry (1986). **The Arpanet an Computer Networks. Proceedings of the Conference on the History of Personal Workstation ,** New-York : ACM, 51-58

- Rosenthal, Dave & Yen, Albert (1983). **User Interface Models Summary in Graphical Input Interaction Technique : Workshop summary**
Computer Graphics; January 1983, pp 16-20

- Rubinstein, Richard & Hersh, Harry M. (1984). **Design Philosophy**
Chapter 2 in The Human Factor : Designing Computer System For People
Burlington, MA : Digital Press, pp 12-22

- Sackman, Harold (1970). **Man-Computer Problem Solving : Experimental Evaluation of Time-Sharing and Batch Processsing,**
Princeton : Awerbach Publishers

- Schmucker, Kurt (1986). **MacApp : an Application Framework,**
in Baecker, R.M. & Buxton W.A.S. Readings in Human-Computer Interaction
A Multidisciplinary Approach, Morgan Kaufmann Publishers, Inc.95 First Street, Los Altos, California 94022

- Schneiderman, Ben (1983). **Human Engineering Management Plan For Interactive Systems**
Proceedings of the IEEE Compcon 83 ConferenceWashington, D.C., September 1983, pp 230-238

- Schneiderman, Ben (1987). **Designing the User Interface : Strategies for Effective Human-Computer Interaction**
Addison-Wesley Publ. Comp., 1987

- Shohmeier, Alfred (1986). **Le materiel informatique : concepts et principes**
Presses Polytechniques romandes, Lausanne, 1986

-. Sproull, (1983). **Programming the User Interface in Man-Machine Interaction,** Proceedings of the joint Conference
IBM/University of Newcastle upon Tyne Computing Laboratory, september 1983

- Sufrin, Bernard (1986). **Formal Methods and the Design of Effective User Interface**
in People and Computers : Designing for Usability, Proceedings of the Second Conference of the Britisch Computer Society Human Computer Interaction Specialist Group, Britisch Computer Society Workshop Series

- Sutherland, I.E. (1963). **Sketchpad : A Man-Machine Graphical Communication System**
AFIPS Conference Proceedings 23, 329-346

- Tanner, P., Buxton, W. (1983). **Some Issues in Future User Interface Management Systems (UIMS) Development**, IFIP Working Group 5.2. Workshop on User Interface Management, Secheim West Germany, Novemeber 1983

- Tardelli, Peter & Cooper Paul (1986). **Design and Evaluation of the Aide Adaptative Front-end to Telecom Gold**, in People and Computers : Designing for Usability, Proceedings of the Second Conference of the Britisch Computer Society Human Computer Interaction Specialist Group, Britisch Computer Society Workshop Series

- Thacher, C.P., McCreight, E.M., Lampson, B.W., Sproull, R.F. & Boggs, D.R. (1979). **Alto : A Personal Computer**. In Siewiorek, Bell, & Newell, Computer Structures. Principles and Examples, Second Edition, Mc Graw-Hill, 549-572

- Wasserman, A.I. (1986). **Extending State Transition Diagrams for the Specification of Human-Computer Interaction**, in Baecker, R.M. & Buxton W.A.S. Readings in Human-Computer Interaction A Multidisciplinary Approach, Morgan Kaufmann Publishers, Inc.95 First Street, Los Altos, California 94022

- Wasserman, A.I., Pircher, P.A. , Shewmake, D.T. & Kersten, M.L. (1986) . **Developping Interactive Information Systems with the User Software Engineering Methodology** IEEE Transaction on Software Engineering SE-12(2), February 1986, pp 326-345

- Weinberg, Gerald (1971). **The Psychology of Computer Programming**, New-York : Van Nostrand Reinhold

- Williams, Gregg (1984). **The Apple Macintosh Computer.** Byte 9 (2), February 1984, 30-54